Universidad
Carlos III de Madrid

Ingeniería Técnica en Informática de Gestión
Departamento de Informática

PROYECTO FIN DE CARRERA

# ADGSO - Aplicación con fines didácticos para la creación y carga de datos espaciales en Oracle

Autor: Esther Herva Landeira

Tutor: Lourdes Moreno López

Leganés, julio de 2011

Título: ADGSO – APLICACIÓN CON FINES DIDÁCTICOS PARA LA CARGA Y CREACIÓN DE DATOS ESPACIALES EN ORACLE
Autor: ESTHER HERVA LANDEIRA
Director: LOURDES MORENO LÓPEZ

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

Me gustaría aprovechar la oportunidad para mostrar mi agradecimiento a todas aquellas personas que han contribuido a que hoy esté escribiendo estas palabras.

A Carlos y Rosa por su apoyo y esfuerzo a los largo de todos estos años. Gracias por ayudarme a llevar a buen puerto el proyecto de labrarme un futuro profesional.

A Gonzalo, por ayudarme, apoyarme y animarme en todo momento. Gracias por decantar el reparto de tareas a mi favor.

A mis compañeros de carrera que consiguieron amenizar el camino. Me llevo muy buen recuerdo de las largas tardes de estudio en la cafetería y las sesiones de prácticas acompañadas de una buena dosis de tabletas de chocolate.

A Lourdes, por ser una tutora de proyecto accesible y cercana que me ha brindado siempre su ayuda, compresión y paciencia.

Gracias a todos.

# Resumen

Desde la aparición de los primeros SIG en los años 1960 y 1970 cada proveedor ha usado modelos específicos de datos y diferentes métodos de almacenamiento de los mismos.

La diversidad de modelos y formatos específicos de cada SIG y la separación de la información espacial del resto de datos, ha representado un obstáculo para el pleno despliegue del valor añadido de los datos espaciales en las organizaciones. Con el crecimiento del uso de los SIG en las empresas y en el sector público, algunas de sus limitaciones se han hecho evidentes. Las organizaciones a menudo tienen que tratar con múltiples estándares e incompatibilidades para el almacenamiento de datos espaciales, además de utilizar distintos idiomas e interfaces para analizar los datos.

Los SGBD con el fin de aprovechar la naturaleza de una base de datos relacional y eliminar la separación de los datos espaciales y no espaciales, en los últimos 5 a 10 años han incorporado extensiones espaciales, que incluyen esquemas de almacenamiento de la información espacial. Una vez que los datos espaciales se almacenan en una base de datos, pueden tratarse, recuperarse y relacionarse como el resto de datos.

Las soluciones aportadas por los SGBD en la actualidad constituyen plataformas con una amplia gama de herramientas para la gestión de la información espacial.

Oracle Spatial, solución aportada por uno de los sistemas gestores de bases de datos de uso más extendido en la actualidad, constituye una de las soluciones más completas del mercado. El proyecto se centra en esta solución, concretamente en llevar a cabo un estudio sobre cómo se gestiona la información espacial en una base de datos con integridad referencial.

En primer lugar como introducción se dará una visión general de la importancia del estudio del dato espacial, indicando los ámbitos de uso más frecuente y las soluciones que se aplican en dichos ámbitos para tratar la información espacial.

Seguidamente se dará una visión detallada del producto Oracle Spatial, cubriendo los aspectos más básicos e importantes.

Por último, como culminación de la consecución del objetivo del proyecto se describirá el análisis, diseño e implementación del aplicativo ADGSO, herramienta de ayuda y asistencia en el aprendizaje del almacenamiento de datos de tipo espacial en Oracle.

# Abstract

Since the first appearances of GIS in 1960 and 1970, each provider has used specific data models and different methods to store them.

The diversity of models and specific GIS formats, and the separation of spatial information from other data, has represented an obstacle to unlock the value of spatial data in organizations. With the growing use of GIS in business and the public sector, some of the limitations of GIS have become apparent. Organizations often have to deal with multiple standards and incompatibilities for storing spatial data, in addition to use different languages and interfaces for data analysis.

DBMS to take advantage of object relational data model and eliminate the separation between spatial and non spatial data in the last 5 to 10 years have incorporated spatial extensions, which include storage schemes of spatial information. Once the spatial data is stored in a database can be manipulated, recovered and relate as all other data stored in the database.

The solutions provided by DBMS at the present are truly a complete geospatial data management platform with a wide range of tools to manage spatial information.

Oracle Spatial the solution provided by one of the management systems databases most commonly used today, is one of the most complete solutions on the sector. The project focuses on this solution. It will conduct a study on how to manage spatial information in a relational database.

First, as introduction it will give an overview about the importance of studying the spatial data, indicating the areas most frequently used and the solutions applied in these areas to manipulate spatial information.

Next it will give a detailed overview of Oracle Spatial product, covering the most basic and important aspects.

Finally as the culmination of project objectives it will describe the analysis, design and implementation of ADGSO application, supporting and assisting tool in learning to storage the spatial data types in Oracle.

# Índice general

# Índice de figuras

ÍNDICE DE FIGURAS

# Índice de tablas

# Capítulo 1

# Introducción y objetivos

## 1.1 Introducción

Como se verá más adelante en la presentación del estado de la cuestión, la información espacial ya no es solo objeto de estudio en áreas de aplicación especializada, cada vez está más presente en la vida cotidiana. Día a día se utilizan herramientas de análisis de información espacial, que resuelven preguntas frecuentes como ¿Dónde se encuentra algo o alguien?, ¿Qué hay a alrededor de interés?, ¿Cómo se llega a un lugar determinado?, ¿Qué ruta es la más corta?, ¿Cuál es la predicción meteorológica en la ubicación actual?, etc.

Cada vez son más evidentes las ventajas de explotar la información espacial en cualquier ámbito, por lo que en la actualidad hay una gran oferta de software dedicado al manejo de la información espacial.

En apartados posteriores se verá que las herramientas guardan la información espacial en diferentes soportes. Los sistemas gestores de bases de datos están aportando soluciones cada vez más completas para el manejo del dato espacial. Este hecho no pasa desapercibido entre las empresas dedicadas al desarrollo de herramientas de sistemas de información geográfica. La posibilidad de mantener toda la información, tanto espacial como no espacial, en una base de datos con integridad referencial, en lugar de mantener la información espacial separada del resto y establecer un mecanismo de vinculación propio del sistema de información geográfica, puede ofrecer grandes ventajas comó se verá en detalle más adelante.

Oracle aporta una de las soluciones más completas con Oracle Spatial. El proyecto se centra en esta solución, concretamente en cómo se gestiona la información espacial en una base de datos con integridad referencial de Oracle.

Al adentrarse en el estudio de Oracle Spatial, se observa cierto grado de dificultad a la hora de generar un script que permita cargar escenarios de trabajo con información espacial. Esta dificultad es la que ha motivado la creación de la aplicación didáctica ADGSO, que proporcionará al usuario una interfaz sencilla que le permitirá crear capas, introducir datos espaciales en las mismas y validar constructores de datos espaciales; a la vez que presentará las sentecias SQL utilizadas y las explicaciones pertinentes en cada caso, de modo que sirva de aprendizaje y apoyo en el conocimiento de Oracle Spatial.

# 1.2 Objetivos

El propósito principal de este proyecto es llevar a cabo un estudio sobre cómo se maneja la información espacial en un sistema gestor de bases de datos, concretamente en Oracle Spatial, solución aportada por uno de los sistemas gestores de bases de datos más utilizados. Para ello, se va a desarrollar la aplicación didáctica ADGSO, que va a servir como herramienta de ayuda y asistencia en el aprendizaje del almacenamiento de datos de tipo espacial, además de proporcionar una forma sencilla de crear escenarios de trabajo en dos dimensiones que sirvan de base de estudio para el análisis, manipulación y visualización de datos espaciales.

Para cumplirlo, se van a establecer los siguientes objetivos:

- Ofrecer una visión general de la importancia del estudio del dato espacial, indicando los ámbitos de uso más frecuente y las soluciones que se aplican en dichos ámbitos para tratar la información espacial.

- Realizar un estudio exhaustivo de la solución aportada por Oracle para la gestión de la información espacial. Se dará una visión detallada del producto, desde los componentes de su arquitectura, hasta adentrarse en los detalles de diseño de la información espacial, su estructura lógica y física, sus metadatos, los métodos de carga, validación y depuración de datos, la indexación y análisis de la información espacial y por último la visualización y definición de mapas.

- Una vez establecidas las bases de conocimiento, los objetivos del proyecto se centran en el desarrollo de la aplicación didáctica ADGSO. Para ello es necesario:

    - Analizar y describir la especificación de requisitos software que debe satisfacer el aplicativo ADGSO.

    - Diseñar la estructura y componentes del nuestro sistema ADGSO necesarios para cumplir los requisitos establecidos en el análisis de las necesidades del aplicativo.

- Implementar la solución utilizando las herramientas y técnicas de programación adecuadas escogidas para el desarrollo.

- Por último, instalar el aplicativo y verificar que el sistema es correcto.

# 1.3 Fases del desarrollo

El proyecto se va a llevar a cabo en tres fases principales que se desarrollarán a lo largo de los siguientes apartados:

- Investigación sobre las bases de datos espaciales y en concreto sobre la solución aportado por el SGBDR Oracle.

- Análisis, diseño e implementación del aplicativo ADGSO como herramienta didáctica de ayuda y asistencia en el aprendizaje del almacenamiento de datos de tipo espacial.

- Instalación, verificación y puesta en funcionamiento del aplicativo ADGSO.

# 1.4 Medios empleados

Para la realización del proyecto se ha contado con los recursos aportados desde el Laboratorio de Bases de Datos Avanzadas (LaBDA)[1] del departamento de Informática de la Universidad Carlos III de Madrid al que mi tutora pertenece, además de un equipo personal para poder trabajar en él de manera autónoma en ocasiones. Las telecomunicaciones, software y hardware utilizado se describen en el capítulo siete para la definición de presupuesto.

# 1.5 Estructura de la memoria

En este apartado se pretende ofrecer una breve visión global de los contenidos expuestos en este documento. Se explicará de forma resumida cada uno de los capítulos que conforman el escrito:

- Capítulo 1 'Introducción y Objetivos': se trata de una introducción general del proyecto, donde en primer lugar se indican las motivaciones que han llevado al desarrollo del proyecto, seguidamente se fijan los objetivos

[1] Grupo LaBDA: http://labda.sintonia.inf.uc3m.es

marcados y por último se da una visión a grandes rasgos de los temas que se van a tratar en cada capítulo.

- Capítulo 2 ‘Estado del Arte’: se expone el estado de la cuestión, es decir los trabajos relativos y fundamentos teóricos que han dado lugar al desarrollo del proyecto.

- Capítulo 3 ‘Oracle Spatial’: se entra en detalle en la solución aportada por Oracle para dar soporte al almacenamiento y manejo de información espacial, Oracle Spatial, que es el objeto de estudio del proyecto.

- Capítulo 4 ‘Aplicación’: incluye la documentación referente al desarrollo del proyecto ADGSO. Ha sido generada en base al estándar de la ‘IEEE 1058.1 – 1987’ para el desarrollo de planes de proyectos software.

- Capítulo 5 ‘Documentación’: recoge los manuales tanto de administración como de uso del aplicativo ADGSO.

- Capítulo 6 ‘Conclusiones’: se exponen las conclusiones extraídas a lo largo del desarrollo del proyecto y se proponen desarrollos futuros para mejorar los resultados en proyectos o investigaciones posteriores.

- Capítulo 7 ‘Presupuesto’: contiene la evaluación del presupuesto total del desarrollo del proyecto, indicando un desglose de costes de personal, costes de material y costes totales.

- Capítulo 8 ‘Glosario’: catálogo de los principales términos que aparecen a lo largo del documento y su definición para ayudar al entendimiento del mismo.

- Capítulo 9 ‘Referencias’: contiene la evaluación de las referencias a las fuentes de información consultadas para la realización del proyecto.

# Capítulo 2

# Estado del Arte

## 2.1 Información espacial

La información espacial en términos simples, es el conocimiento sobre donde se encuentra un elemento con respecto a un sistema de referencia, sobre donde está una cosa o que hay en un determinado lugar.

El hombre es un ser espacial, las referencias del espacio forman parte de la vida cotidiana. La frase “todo lo que pasa, pasa en algún lugar” puede parecer una obviedad, pero es realmente importante. La información espacial forma parte de la vida del ser humano de una manera cotidiana y muy constante, permitiendo dar solución a problemas encontrados en el día a día. Se puede interpretar un mapa de metro para decidir el recorrido a seguir, guiarse en el desplazamiento con el coche gracias a los mapas de un navegador, dibujar un plano para indicar la distribución de una habitación, escoger la ruta más corta de llegada al trabajo etc.

Pero más allá de las cuestiones más cotidianas, el campo de aplicación de la información espacial es muy amplio y va en aumento. El uso que se hace de ella, está adquiriendo un papel estratégico en nuestra sociedad dado el gran potencial que aporta al servicio de los ciudadanos.

A continuación, se puede observar una clasificación de los ámbitos principales de aplicación:

- TRANSPORTES

  El reto de las agencias de transporte público consiste en prestar el mejor servicio posible al ciudadano. El análisis de la información espacial se aplica como apoyo a actividades tales como:

  - Gestión de las infraestructuras de transporte (Mantenimiento y Conservación).
  - Generación de rutas optimas en función de condiciones.
  - Evaluación de los impactos territoriales causados por la construcción de nuevas infraestructuras.
  - Realización de simulaciones sobre el comportamiento del tráfico, densidades de circulación etc.
  - Análisis de tránsito de viajeros.
  - Estudios de accesibilidad sobre los efectos territoriales de las redes de transporte a distintas escalas espaciales: europea, nacional y metropolitana. Así como sobre los efectos de planes de infraestructuras, líneas de alta velocidad y autopistas de circunvalación.

  La gestión efectiva de las actividades anteriores contribuye a mejorar el servicio al ciudadano influyendo directamente en prestaciones tales como minimización de tiempos de espera, distribución equitativa de paradas, disponibilidad de transporte en una zona determinada, generación de rutas, actuaciones en cortes de rutas de transportes, etc.

- ADMINISTRACIÓN PÚBLICA

  Las Administraciones Públicas puedan abordar sus proyectos con la componente espacial en toda la información disponible, y así maximizar su uso en todos los procesos de toma de decisiones, tales como:

  - Gestión de Catastro.
  - Mantenimiento y Control Urbanístico.
  - Gestión de Obras Públicas.
  - Soluciones para Medio Ambiente y Patrimonio Verde.
  - Gestión Sanitaria.
  - Servicios Sociales.
  - Estadística y Padrón

- Servicios y difusión de la información al ciudadano (guías urbanas, callejeros, interacción con la administración, etc.)

Conocer la localización de la población o un determinado segmento de la misma provoca un impacto directo en el grado de efectividad a la hora de aportar soluciones tales como ubicar un nuevo centro social, distribuir los servicios sanitarios de forma equitativa en un territorio, aportar servicios de callejero al ciudadano, realizar un análisis de los resultados electorales por comunidades, etc.

- RECURSOS NATURALES

El análisis espacial permite realizar una gestión de la explotación más efectiva y en ocasiones con menos impacto ambiental, en recursos naturales:

- Agricultura.
- Cambio Climático.
- Gestión Forestal.
- Gestión Medioambiental.
- Hidrología - Confederaciones Hidrográficas.
- Mapas de ruido.
- Medio Marino.
- Minería.
- Petróleo.

El uso de la información espacial sirve de apoyo a actividades tales como identificación de terrenos idóneos en un uso agrícola determinado, localización de zonas afectadas por el cambio climático, generación de cartografía de zonas inundables, estudios sísmicos, etc.

- SEGURIDAD Y EMERGENCIAS

Los servicios de seguridad y emergencias se enfrentan en el día a día a situaciones peligrosas que requieren una actuación rápida y eficaz:

- Bomberos
- Centros de coordinación de emergencias
- Policía Local
- Protección Civil

El análisis de la información espacial se aplica en base a aumentar la rapidez de respuesta, realizar estudios de la información y establecer planes de contingencia en base a dichos estudios. La clave de la gestión de las emergencias es la localización eficiente de las incidencias que redunda en una gestión eficiente de las emergencias. Sirve de apoyo de actividades como generación de planos de zonas de riesgos como gasolineras o fabricas de químicos, simulación de incendios para detección de rutas de escape, localización de tramos de tasa alta de accidentes de tráfico, análisis y vinculación de delitos con sospechosos, etc.

- SERVICIOS EMPRESARIALES

En el ámbito empresarial cada día cobra más importancia el análisis y explotación de la información espacial, es un mundo muy competitivo en el cual es necesario encontrar un elemento diferenciador para posicionarse en el mercado mejor que cualquier otro competidor:

  - Banca y Servicios Financieros
  - Gestión de infraestructuras
  - Inmobiliaria
  - Medio de comunicación y prensa
  - Seguros
  - Venta al por menor

El análisis de la información espacial ayuda a mejorar la comprensión del riesgo, la interacción con el cliente y las condiciones económicas con modelos espaciales basados en la geografía.

La mejor manera de conocer a los clientes y hacer crecer la empresa es mediante el análisis del mercado y la combinación de la información de la empresa con los datos de localización.

Los sistemas de información espacial potencian el ahorro de costes y provocan una mayor eficiencia mediante la automatización, la mejora de los flujos de trabajo y la toma de decisiones basada en el intercambio de conocimientos de forma rápida e intuitiva.

- OTROS

Además de los ámbitos de aplicación anteriores en los que los sistemas de información geográfica ya están totalmente implantados, están surgiendo nuevos ámbitos de aplicación motivados por las grandes ventajas que pueden aportar los sistemas de información geográfica, tales como:

  - Inteligencia Geoespacial
  - Instalaciones militares

- Operaciones Militares - C4IS.

En los siguientes apartados se verán algunos de los sistemas de información geográfica más importantes y las herramientas de uso más extendido.

# 2.2 Soluciones en el mercado

En el apartado anterior se han mencionado la gran variedad de ámbitos en los que tiene aplicación el uso de la información espacial, así como los diferentes problemas abarcados en cada uno de ellos. Cabe preguntarse ¿cómo se encuentran las respuestas a dichos problemas? Utilizando lo que se tradicionalmente se denominan Sistemas de Información Geográfica o SIG.

Existen muchas definiciones de SIG, algunas de ellas acentúan su componente de base de datos, otras sus funcionalidades y otras enfatizan el hecho de ser una herramienta de apoyo en la toma de decisiones, pero todas coinciden en que se trata de un sistema integrado para trabajar con información espacial, herramienta esencial para el análisis y toma de decisiones en muchas áreas del conocimiento.

Simplificando, se puede decir que un SIG es una integración organizada de hardware, software y datos, diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada, con el fin de satisfacer múltiples propósitos, como los indicados en el apartado anterior.

En general, el correcto funcionamiento de un SIG requiere la integración de cinco componentes clave:


*Figura 1. Componentes de un SIG [ESRI]*

- Hardware

  Los SIG corren en un amplio rango de tipos de computadoras, desde equipos centralizados hasta configuraciones individuales o de red. Una organización requiere de hardware suficientemente específico para cumplir las

necesidades de la aplicación. Algunas cosas a considerar incluyen: velocidad, costo, soporte, administración, escalabilidad y seguridad.

- Software

  Los programas SIG proveen las herramientas y funcionalidades necesarias para capturar, almacenar, manipular, analizar y mostrar la información espacial. En la actualidad de dividen fundamentalmente en:

  - *SIG de escritorio*: son aquellos que se utilizan para crear, editar, administrar, analizar y visualizar los datos espaciales.
  - *Sistemas de gestión de bases de datos espaciales (SGBD espacial)*: se emplean para almacenar la información geográfica, pero también proporcionan otras funcionalidades, tales como el análisis y la manipulación de los datos.
  - *Servidores cartográficos*: se utilizan para distribuir mapas a través de Internet.
  - *Servidores SIG*: proporcionan la capacidad para crear, administrar y distribuir los servicios SIG.
  - *Clientes Web SIG*: permiten la visualización de datos y acceder a funcionalidades de análisis y consulta de servidores SIG a través de Internet o Intranet.
  - *SIG móviles*: se usan para la recogida de datos en campo. Son dispositivos móviles (tales como Tablet PC, sistemas montados en vehículos, teléfonos Windows Smartphone y dispositivos Apple) para ver, recopilar y actualizar la información geográfica.

  Actualmente la mayoría de los proveedores de software SIG distribuyen productos fáciles de usar.

- Datos

  El componente más importante para un SIG es la información. Se requiere de adecuados datos de soporte para que el SIG pueda resolver los problemas y contestar a las preguntas de la forma más acertada posible. Los datos geográficos se pueden comprar a proveedores que se dedican a su distribución u obtenerse mediante recursos propios, haciendo uso de herramientas de escritorio y dispositivos de captación, tales como escáneres, tablas digitalizadores, fotografía aérea, imágenes de satélite, etc.

- Recurso Humano

  Las tecnologías SIG son de valor limitado si no se cuenta con el personal cualificado en manejar el sistema y desarrollar planes de implementación del mismo. Sin el personal experto en su desarrollo, la información se

desactualiza y se maneja erróneamente, lo que deriva en no utilizar todo el potencial del hardware y el software.

- Procedimientos

  Como todo sistema dentro de una empresa, para que un SIG sea exitoso, debe basarse en un buen diseño y reglas de actividad definidas, que son los modelos y prácticas operativas exclusivas de cada organización.

Los resultados obtenidos por el SIG dependen de un buen engranaje y funcionamiento de todos los componentes del mismo, que redundará en una buena administración y gestión de los datos, que como hemos mencionado son la base de la resolución de los problemas.

Por datos, no se debe entender solo datos espaciales, sino también los atributos no espaciales de los mismos, lo que se denominan los datos alfanuméricos. Por ejemplo para crear un mapa que refleja la densidad de población en las provincias de España, se necesita, por un lado los objetos geométricos que representan los límites de las provincias (datos espaciales) y por otro el número de habitantes asociado a casa provincia (datos alfanuméricos).

Desde la aparición de los primeros SIG en los años 1960 y 1970 cada proveedor ha usado modelos específicos de datos y diferentes métodos de almacenamiento de los mismos, como una colección de ficheros organizados en un sistema de archivos, como una colección de tablas dentro de un sistema de gestión de bases de datos relacionales (SGBDR) o una mezcla de ambos. Un sistema muy utilizado y presente todavía en muchos sistemas, consiste en almacenar los datos correspondientes a los elementos gráficos en ficheros y los datos alfanuméricos en tablas de bases de datos, estableciendo la relación entre ellos con un mecanismo de vinculación de identificadores propio del SIG. Este método requiere el manejo de la información geográfica separado del resto de la información almacenada en la base de datos y surgió en sus inicios principalmente como resultado de las limitaciones de la tecnología.

La diversidad de modelos y formatos específicos de cada SIG y la separación de la información espacial del resto de datos, ha representado un obstáculo para el pleno despliegue del valor añadido de los datos espaciales en las organizaciones. Con el crecimiento del uso de los SIG en las empresas y en el sector público, algunas de sus limitaciones se han hecho evidentes. Las organizaciones a menudo tienen que tratar con múltiples estándares e incompatibilidades para el almacenamiento de datos espaciales, y tienen que utilizar distintos lenguajes e interfaces para analizar los datos. Además, sistemas como el CRM (Customer Relationship Management), ERP (Enterprise Resource Planning), o los sistemas utilizados en logística dependen cada vez más de la integración de la información espacial con todos los otros tipos de información. Esto ha sido a menudo un desafío operacional y técnico, que en algunos casos se resolvió de forma manual para obtener información de un sistema y cargarlo en otro para realizar el análisis espacial necesario.

Los SGBDR con el fin de aprovechar la naturaleza de una base de datos relacional y eliminar la separación de los datos espaciales y no espaciales, en los últimos 5 a 10 años han incorporado extensiones espaciales, que incluyen esquemas de almacenamiento de la

información espacial. Una vez que los datos espaciales se almacenan en una base de datos, puede tratarse, recuperarse y relacionarse como el resto de datos. La integración de la información espacial con el resto de información no espacial, aporta muchas ventajas tales como, el acceso de múltiples usuarios concurrentes a conjuntos de datos contiguos, la gestión de datos espaciales y de negocios en un entorno integrado, los índices espaciales, control de versiones, la seguridad, y apoyo a las funciones inteligentes.

Las soluciones aportadas por los SGBDR han mejorado mucho desde sus primeras versiones y en la actualidad constituyen plataformas con una amplia gama de herramientas para la gestión de la información espacial. Las más conocidas e instauradas en el mercado en la actualidad son Oracle 11G R2, SQL Server 2008 R2 y como opción de software libre PostGIS 1.5.

Todos ellas siguen las especificaciones del Open Geospatial Consortium (OGC), dedicado a la definición de estándares abiertos e interoperables dentro de los SIG, ya sea adaptándose de modo estricto a los formatos del estándar 'SQL/MM', con sus propios tipos de datos esencialmente equivalentes a los formatos del estándar 'SQL/MM' u ofreciendo ambas posibilidades.

La ventaja a destacar es que los conocimientos que se adquieren en cualquiera de los productos son bastante aplicables a los demás.

## 2.3 Herramientas

En el apartado anterior se ha visto que uno de los componentes clave de un SIG es el software. En la actualidad hay una gran variedad de herramientas disponibles en el mercado, algunas de las más conocidas e instauradas, clasificadas por empresa, son:

- ESRI: familia de productos ArcGIS
- Bentley: MicroStation, Bentley MAP, Bentley GeoSpatial Server, Bentley Geo Web Publisher.
- Autodesk: AutoCAD, Autodesk Map.
- MapINFO: MapINFO Professional.
- Geomedia: Geomedia, Geomedia Web Map.
- Software libre: gvSIG, GRASS, Kosmo.

El mercado de aplicación de estas herramientas es muy amplio, algunas de ellas se han orientado más a lo puramente SIG, como ESRI, mientras que otras han explotado últimamente el mercado de diseño y la Ingeniería y como un plus ha incursionado en SIG pero con un enfoque de sus clientes existentes como Bentley.

Para dar una visión general de lo que aportan las herramientas SIG, se va a presentar la solución aportada por la empresa ESRI, empresa líder mundial en el sector de desarrollo y comercialización de los sistemas de información geográfica.

## 2.3.1 ArcGIS como ejemplo de herramientas SIG

A continuación se van a describir los diferentes productos de la familia ArcGIS y el papel de cada uno de ellos dentro del sistema general.

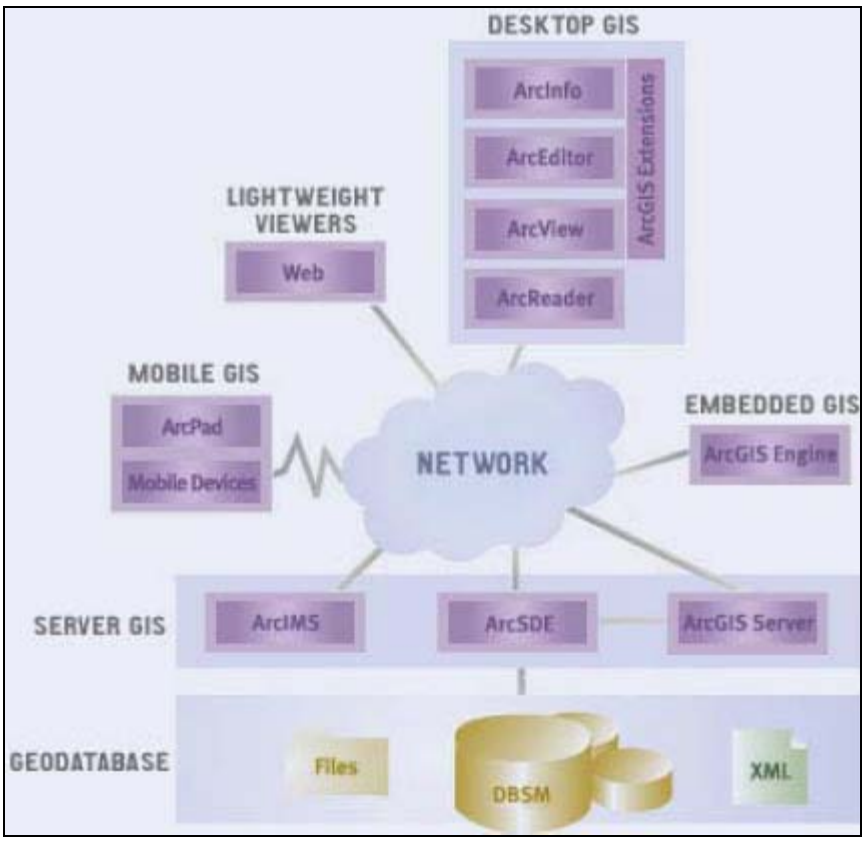



*Figura 2. Familia de productos ArcSIG [ARCGIS]*

### 2.3.1.1 ArcGIS Desktop

Es la familia de aplicaciones SIG de escritorio, es una de las más ampliamente utilizadas. Estas herramientas permiten realizar tareas SIG tales como mapeo, administración de datos, análisis espacial, edición de datos, geoprocesamiento, etc.

La mayoría de los usuarios que utilizan estas herramientas realizan tres tareas clave: trabajar con mapas, realizar análisis espacial y compilar datos. Los mapas son esenciales en Desktop, porque hacen que toda la información cobre vida y son el mecanismo utilizado para editar y proporcionar análisis espacial a los usuarios.

En sus últimas ediciones AcrGIS Desktop incluye las herramientas:

- ArcCatalog: constituye un avanzado explorador de datos geográficos y alfanuméricos, pensado para la visualización, administración y documentación de la información. Permite administrar los archivos del SIG; es el equivalente del explorador de Windows para archivos geográficos. Con el uso de este módulo se facilitan las tareas de renombrar, copiar, borrar, crear nuevas capas y exportar los archivos SIG.

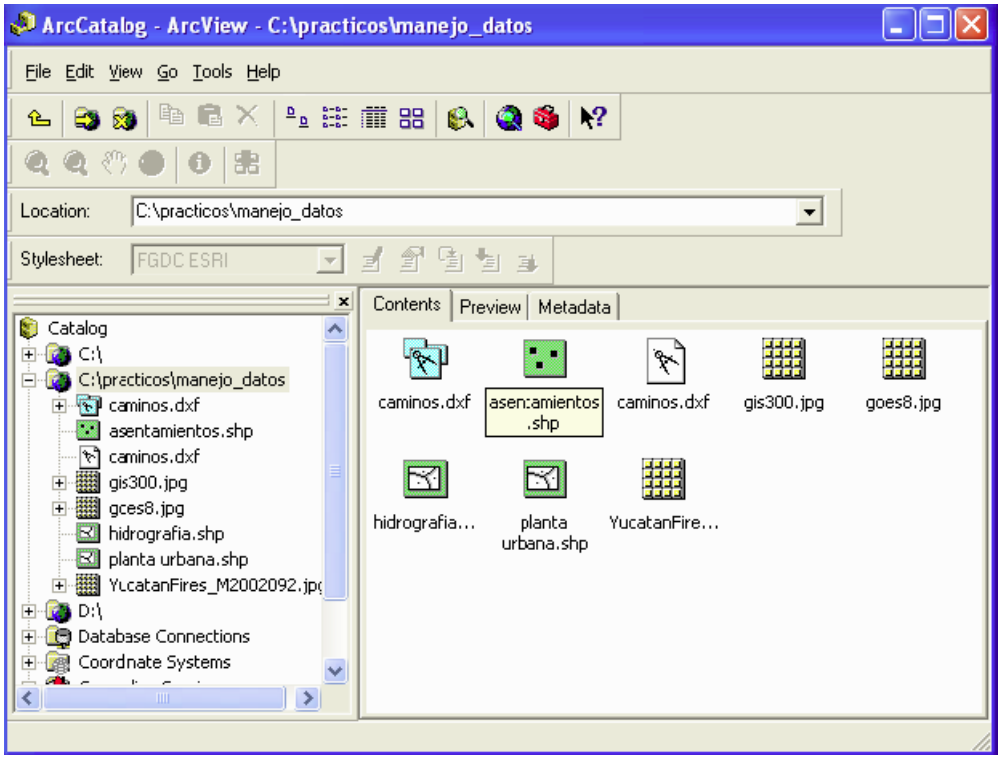



*Figura 3. Interfaz de ArcCatalog*

- ArcMap: es la aplicación central de ArcGIS. Este módulo permite la visualización, creación, edición, consulta, análisis y presentación de los datos. Haciendo uso de las barras de herramientas y los menús contextuales el usuario trabaja de forma visual sobre las capas que forman el mapa. Podrá añadir y eliminar capas, cambiar el orden de despliegue de las mismas en el mapa, modificar sobre el mapa las características espaciales de la información, acceder, añadir y modificar información alfanumérica a las capas, incluir análisis cuantitativos y cualitativos de los datos de capa, etc.

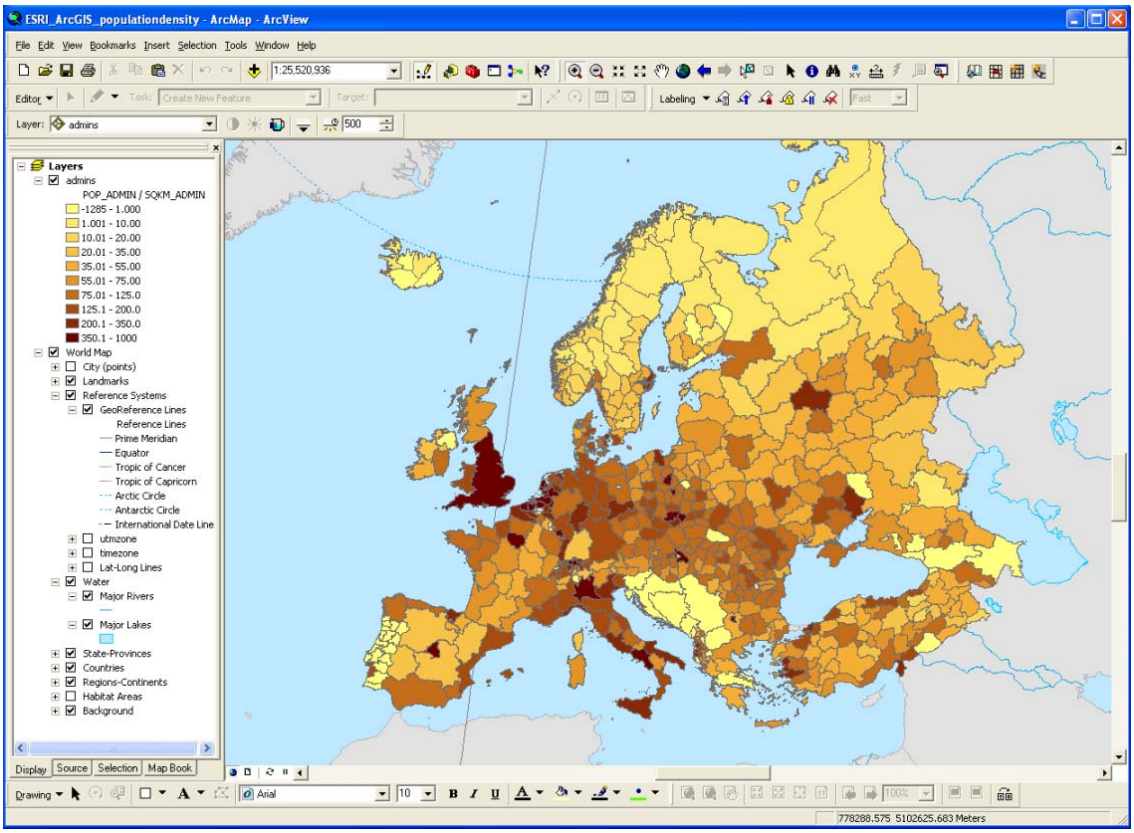



*Figura 4. Interfaz de ArcMap*

- ArcToolbox: es una interfaz que permite acceder, organizar y administrar bloques de herramientas de geoprocesamiento, tales como la realización de conversiones entre los diferentes formatos de datos espaciales, cambios de proyección y ajuste espacial, generación de redes geométricas, creación y calibración de rutas, etc.

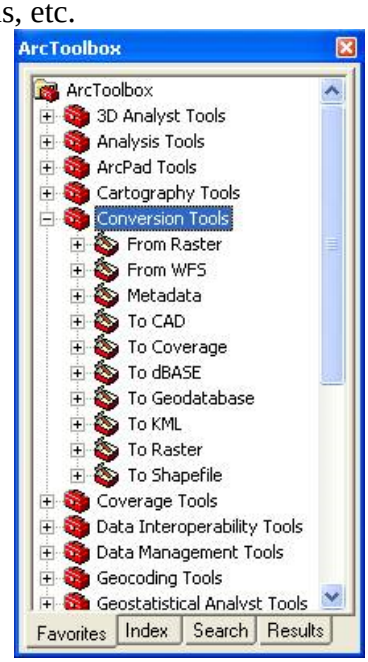



*Figura 5. Interfaz de ArcToolbox*

Estas herramientas, organizados temáticamente y mediante el empleo de intuitivos asistentes, permiten realizar las funciones de forma sencilla e inmediata.

- ArcReader: es una aplicación gratuita que permite a los usuarios visualizar, explorar e imprimir mapas y globos que hayan sido producidos por ArcGIS Desktop y publicados con ArcGIS Publisher. ArcGIS Publisher es una extensión que convierte documentos de ArcMap (.mxd) y ArcGlobe (.3dd) al formato de mapa publicado (.pmf) utilizado con ArcReader.

- ArcScene: es una aplicación de visualización 3D que le permite ver los datos SIG en tres dimensiones. Además proporciona acceso a muchas funciones y herramientas de análisis.

- ArcGlobe: es otra aplicación de visualización 3D que permite visualizar grandes cantidades de datos SIG sobre una superficie del globo. Los datos a los que se hace referencia espacialmente están en una superficie de globo 3D y se visualizan en su ubicación geodésica real. El globo se puede manipular, lo que permite investigar y analizar los datos mientras visualiza el globo como un todo o se acerca a regiones más localizadas.

#### 2.3.1.2 SIG Server

Es una plataforma escalable con tecnología de servidor para crear aplicaciones y servicios SIG profesionales capaces de gestionar, visualizar y analizar información geográfica de forma centralizada.

Permite compartir los recursos SIG en la empresa y en la Web. Estos recursos se comparten al alojarlos en el servidor y permitir a las aplicaciones cliente, ya sean clientes ligeros como navegadores Web o aplicaciones SIG de escritorio, crear y utilizar los recursos.

Las principales ventajas de compartir los recursos SIG en un servidor son en esencia las mismas que las de compartir datos a través de cualquier clase de tecnología del servidor:

- Herramientas que permiten llevar una administración centralizada y crear aplicaciones Web y servicios desde los que los usuario (expertos SIG o no) puedan acceder a toda la funcionalidad SIG disponible, desde un nodo centralizado.

- Integración con otros sistemas de la empresa tales como sistemas de administración de la relación con el cliente (CRM) o el sistema de planeamiento de recursos corporativos (ERP), usando software basado en estándares de la industria. Proporciona las herramientas necesarias para diseñar una Arquitectura Orientada a Servicios (SOA).

- Proporciona escalabilidad. Al ampliar el número de usuarios que acceden al servidor para hacer uso de las aplicaciones SIG disponibles, las posibilidades

del sistema pueden ampliarse aumentando la cantidad de recursos en la máquina servidor o aumentando el número de instancias de ArcGIS Server.

- Permite reducir los costes de una organización, ya que la posibilidad de alojar toda la información SIG empleada a nivel corporativo en el servidor empresarial, hace innecesarios los procesos de instalación de software en todos los equipos de la empresa, así como la duplicación del personal de mantenimiento en el sistema.

- Utilización de estándares, tanto en el sector SIG (OGC) como en el resto de tecnologías de la información (XML, SOA), que permiten la máxima interoperabilidad y compatibilidad con los sistemas empresariales más empleados.

Como componentes de servidor se pueden citar:

- ArcGIS Server: es un completo servidor SIG basado en tecnología Web que proporciona un amplio número de aplicaciones de usuario y servicios para la administración, visualización y análisis espacial de la información. El servidor SIG es el que aloja los recursos SIG, como mapas, herramientas de geoprocesamiento, globos, localizadores de direcciones, etc. y los expone como servicios a las aplicaciones cliente.

- ArcGIS SDE: es la pasarela SIG de a las bases de datos espaciales implementadas sobre los SGBD líderes del mercado como Oracle o Microsoft SQL Server. Es la herramienta proporcionada por ArcGIS para administrar los datos espaciales junto con el resto de datos de la organización. Teniendo en cuenta que el proyecto se centra en el estudio sobre cómo se maneja la información espacial en un sistema gestor de bases de datos, es una herramienta que interesa especialmente y que se desarrollará con más detalle más adelante.

- ArcIMS: constituye el software base para la distribución y difusión de información geográfica, mapas, metadatos y servicios SIG en Internet. Es una solución para la construcción de portales mediante los cuales los usuarios pueden publicar y compartir conocimiento e información geográfica.

#### 2.3.1.3 ArcGIS Explorer

ArcGIS Explorer es un visor ligero de información geográfica que proporciona una manera fácil de explorar, visualizar y compartir información SIG. Es una herramienta gratuita, cuyo principal objetivo es ayudar a la distribución de datos propios de una organización a un público más amplio. ArcGIS Explorer permite fusionar datos locales con servicios de mapas para crear mapas personalizados, añadir fotografías, informes, videos y otra información relevante, realizar preguntas sobre el mapa y compartir los resultados de todas estas operaciones con otros usuarios.

### 2.3.1.4 ArcGIS Engine

ArcGIS Engine es una colección de componentes SIG integrables y recursos que pueden utilizar los desarrolladores para ampliar ArcGIS, añadir capacidades SIG a aplicaciones existentes o crear nuevas aplicaciones a medida. Permite desarrollar, desde una nueva herramienta de visualización de cartografía integrada con otra aplicación, hasta una completa aplicación independiente con funcionalidad SIG avanzada (edición, análisis espacial o geocodificación por ejemplo).

Los desarrolladores usan ArcGIS Engine para implementar datos de SIG, mapas y secuencias de comandos de geoprocesamiento en aplicaciones de escritorio o aplicaciones móviles mediante interfaces de programación de aplicaciones (API) para .NET, Java y C++.

### 2.3.1.5 ArcGIS Mobile

Es el software para la integración de funcionalidad SIG en dispositivos móviles. Los avances en la tecnología SIG e informática móvil permiten a las organizaciones llevar el SIG al campo, interactuar directamente con la información que se necesita ver, capturar y actualizar; y sincronizar los cambios entre el campo y la oficina con facilidad.

Las organizaciones han comenzado a reemplazar los sistemas basados en papel por aplicaciones móviles. La posibilidad de comparar datos espaciales y alfanuméricos almacenados en el sistema con las características reales del terreno, agrega valor a las decisiones de campo y abre un nuevo camino en el mantenimiento de la integridad de la información espacial.

ArcGIS proporciona tres soluciones SIG móviles que abordan tareas de campo simples a complejas en una variedad de marcos:

- ArcGIS Mobile: incluye una aplicación móvil basada en tareas para Windows Mobile y dispositivos de Windows que utilizan una arquitectura de servicios Web para sincronizar la información entre el campo y la oficina.

- ArcPad: está centrada en el mapa y se enfoca en las tareas de campo que requieren herramientas geográficas relativamente simples. Estas tareas típicamente se realizan en equipos portátiles (que ejecutan Microsoft Windows CE o Pocket PC).

- ArcGIS Desktop y ArcGIS Engine: estos productos proporcionan herramientas para SIG móviles de alta calidad con sofisticadas herramientas de representación cartográfica, visualización y edición. Estas soluciones se enfocan en tareas de campo que requieren herramientas geográficas más sofisticadas, que se ejecutan típicamente en Tablet PC de alta calidad. A menudo, las visualizaciones del mapa que se utilizan en el campo en las Tablet PC deben contener información detallada en alta resolución.

## 2.3.2 Hacia las bases de datos relacionales

En el apartado "*2.2 Soluciones en el mercado*" se ha visto la conveniencia de integrar la información espacial con los otros tipos de información almacenados en las bases de datos de las empresas, para solventar algunas de las limitaciones que presentaban los SIG tradicionales.

Para terminar con las herramientas que proporcionan los SIG, es interesante destacar la herramienta de servidor ArcSDE.

ArcSDE constituye la pasarela SIG de ESRI a las bases de datos espaciales, implementadas sobre los SGBDR líderes del mercado como Oracle o Microsoft SQL Server. Gestiona el almacenamiento de elementos espaciales y para almacenar la información geográfica utiliza los tipos espaciales de los SGBDR.

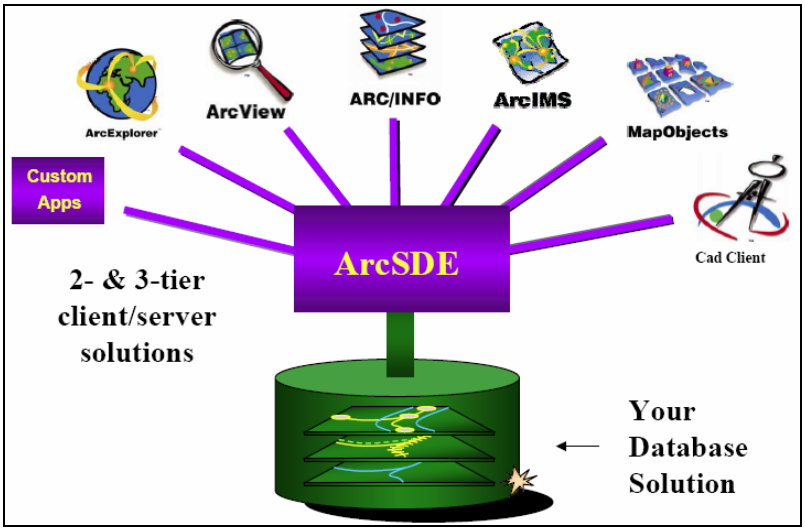



*Figura 6. Arquitectura AcrSDE [ARCGIS]*

Como se puede observar en la figura anterior, es el vínculo necesario entre los SIG y las bases de datos relacionales. Con ArcSDE, los productos SIG (ArcInfo, ArcView GIS, ArcIMS y otras aplicaciones de terceros) acceden directamente a la información geográfica y alfanumérica almacenada en una base de datos relacional, proveyendo entre otras cosas:

- Acceso multiusuario a datos geográficos almacenados en la base de datos relacional.
- Alto rendimiento en el geoprocesamiento de grandes bases de datos.
- Integración con los datos corporativos almacenados en la base de datos.
- Seguridad e integridad de los datos geográficos.

ESRI destaca que ArcSDE, combinado con ArcGIS, provee el ambiente SIG profesional ideal para el mantenimiento de grandes bases de datos geográficas, incluyendo manejo de "versiones" (múltiples ediciones simultáneas), geocodificación e

integración de datos del negocio. Con ArcSDE la información GIS deja de ser un privilegio de los expertos y posibilita que usuarios y aplicaciones de todas las organizaciones accedan y exploten los datos SIG.

Como se ha mencionado en el apartado "*2.2 Soluciones en el mercado*", las organizaciones dedicadas al desarrollo de los SIG, son conscientes de las deficiencias que presentan los modelos tradicionales, de ahí que desarrollen herramientas como ArcSDE , que posibiliten a una organización pasar de una colección de archivos tradicionales basados en los modelos vector, raster, y los datos de diseño asistido por ordenador (CAD) a un entorno integrado, donde todos los datos espaciales y de negocios se gestionan como una base de datos continua.

En el siguiente apartado se verán con más claridad las ventajas que puede aportar el uso de las bases de datos espaciales. Oracle, como plataforma líder en el sector de SGBDR, instaurada en gran cantidad de organizaciones tanto privadas como públicas; y la extensión Spatial, como una de la soluciones más completas y con la que se están integrando los SIG más importantes del mercado, constituye el objeto de estudio de este proyecto y se verá con detalle en el apartado siguiente capítulo. Como introducción, los beneficios a destacar en el uso de Oracle Spatial se resumen a continuación:

- Elimina la necesidad de las arquitecturas duales, como todos los datos pueden ser almacenados de la misma forma. Un almacenamiento de datos unificado significa que todos los tipos de datos (texto, mapas y multimedia) se almacenan juntos, en lugar de que cada tipo se almacene por separado.

- Se utiliza *SQL*, un lenguaje estándar para acceder a bases de datos relacionales, eliminando así la necesidad de lenguajes específicos para manejar los datos espaciales.

- Se define el tipo de datos SDO_GEOMETRY, que es esencialmente equivalente a los tipos espaciales en los estándares *OGC* y SQL/MM.

- Se implementan formatos SQL/MM "bien conocidos" para especificar los datos espaciales. Esto implica que cualquier solución que se adhiere a las especificaciones de SQL/MM pueden almacenar con facilidad los datos en Oracle Spatial y viceversa, sin la necesidad de convertidores.

- Es el estándar de-facto para almacenar/acceder a los datos en Oracle y el intercambio de datos entre aplicaciones por muchos fabricantes, incluyendo NAVTEQ, Tele Atlas, Autodesk, MapInfo, ESRI, Bentley, Intergraph, Radius, Skyline y muchas otras.

- Proporciona escalabilidad, integridad, seguridad, capacidad de recuperación y características avanzadas de administración de usuarios para el manejo de datos espaciales que son la norma en bases de datos Oracle, pero no necesariamente es así en otras administraciones de herramientas espaciales.

- Se elimina la necesidad de organizaciones independientes para mantener una infraestructura de datos espaciales (*hardware*, *software*, soporte, etc.), y se elimina la necesidad de herramientas y habilidades especificas para el funcionamiento de los datos espaciales.

- A través del servidor de aplicaciones, permite a casi cualquier aplicación beneficiarse de la disponibilidad de la información espacial y la inteligencia, reduciendo los costes y la complejidad de las aplicaciones espaciales.

Con Oracle 10g (y sus posteriores versiones), se introdujeron los beneficios de la computación en red para bases de datos espaciales. Para las grandes organizaciones que administran los activos de datos muy grandes, tales como cámaras de compensación o servicios públicos, la flexibilidad y escalabilidad de la red puede suponer un ahorro sustancial de costes y facilitar el mantenimiento de las estructuras de base de datos.

# Capítulo 3

# Oracle Spatial

## 3.1 Introducción

Oracle Spatial, es la solución desarrollada por Oracle para la gestión de bases de datos espaciales. Surgió a partir de la versión 8.1.5 de Oracle siendo, actualmente la versión 11g una versión muy mejorada con respecto a las anteriores, que proporciona una base para aplicaciones SIG complejas.

En la figura "*Figura 7. Arquitectura Oracle Spatial*" se pueden observar los componentes que conforman la arquitectura de Oracle Spatial en su versión 11g.

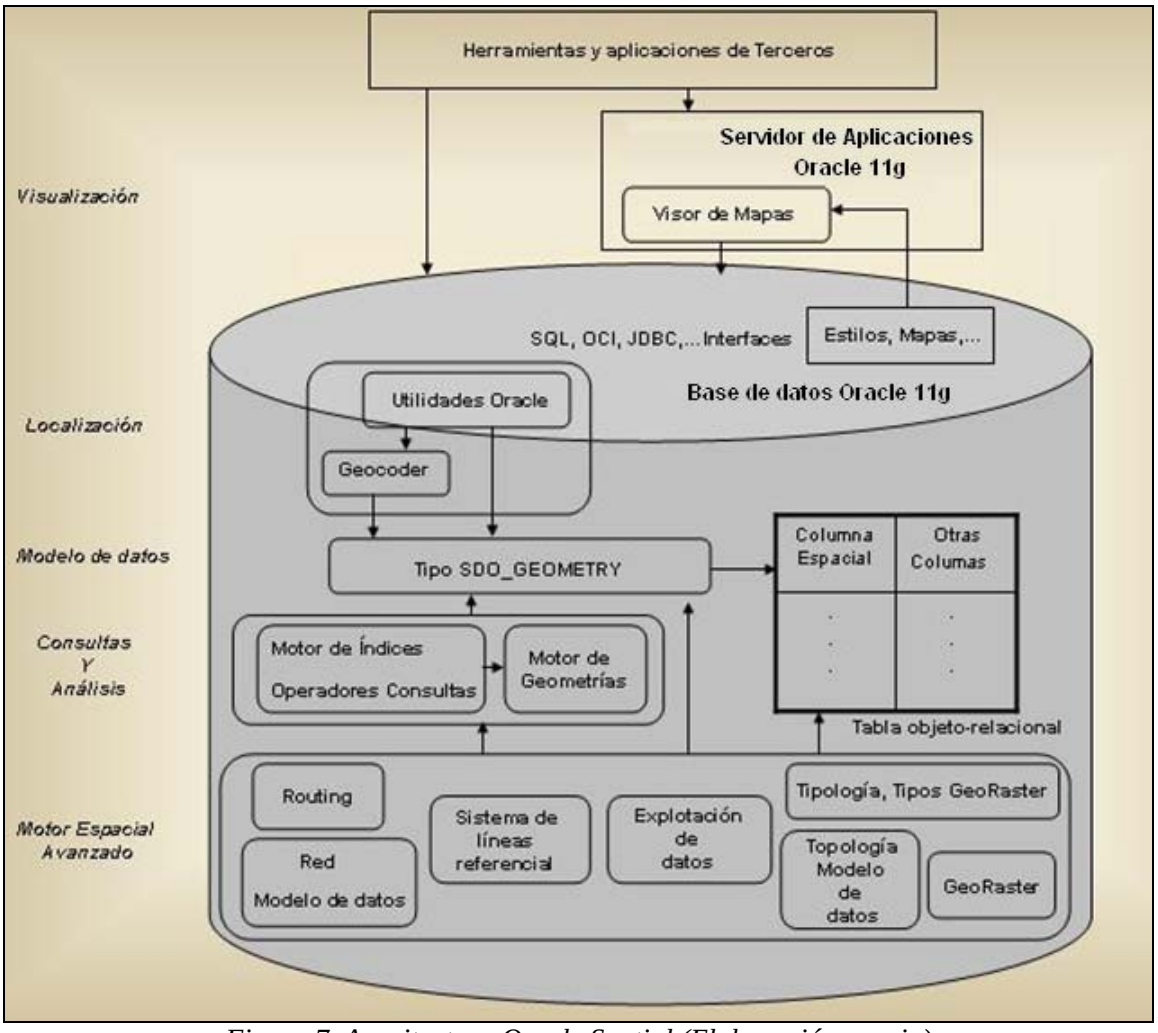



*Figura 7. Arquitectura Oracle Spatial (Elaboración propia)*

Para describir las funcionalidades que permite cubrir Oracle como base de aplicaciones GIS, se va a dar una breve descripción de cada uno de estos componentes:

- Modelo de Datos: Oracle Spatial se apoya en un modelo objeto-relacional para incluir en la base de datos la información geográfica. Este modelo almacena las geometrías en objetos nativos de tipo SDO_GEOMETRY. Los usuarios pueden definir tablas que contengan columnas de tipo SDO_GEOMETRY, para almacenar localizaciones de clientes, almacenes, restaurantes, o incluso, localizaciones y extensiones espaciales de entidades como carreteras, parques, parcelas de tierra, etc.

- Inserción de información espacial: los usuarios pueden añadir información en las columnas de tipo SDO_GEOMETRY, usando utilidades estándar de Oracle tales como, sentencias SQL, importaciones y exportaciones de datos y transformaciones de datos suministrados por fuentes externas, como la conversión de información espacial implícita en columnas SDO_GEOMETRY, usando el componente Geocoder de Oracle Spatial.

- Consultas espaciales y análisis: los usuarios pueden consultar y manipular los datos SDO_GEOMETRY, mediante el motor de geometrías, permitiendo

también el uso de índices espaciales para mejorar los tiempos de respuesta de las consultas.

- Motor Espacial Avanzado: este componente incluye a su vez varios componentes cuya funcionalidad consiste en sofisticar las aplicaciones espaciales, como por ejemplo, dotándolas de SIG y bioinformática. Esto incluye, por citar alguno, el componente GeoRaster, que permite almacenar objetos espaciales usando imágenes (grupos de píxeles), como, puntos, líneas y vértices.

- Soporte de referencia Lineal: Oracle Spatial, soporta el almacenamiento de la información tipo 'medidas' asociadas a una geometría lineal. Esta característica es clave para soportar aplicaciones lineales de red, tales como callejeros en Internet, transporte, redes de telecomunicación, etc.

- Visualización: los componentes del servidor de aplicaciones de la tecnología Oracle Spatial incluyen la forma de visualizar los datos espaciales a través de la herramienta MapViewer (Visor de Mapas). El visor de mapas renderiza los datos espaciales que se encuentran almacenados en las columnas SDO_GEOMETRY, de tablas Oracle y los presenta como mapas.

En la figura "*Figura 7. Arquitectura Oracle Spatial*" se han incluido las herramientas de terceros, que pueden acceder a los datos espaciales, a través de cualquier servidor de aplicaciones, o directamente desde la base de datos usando SQL, OCI, JDBC, o cualquier otro interfaz que se preste a ello.

Las funcionalidades referentes al uso del motor espacial avanzado, solo están incluidas en las ediciones personales y de empresa, en una opción no libre, llamada Spatial.

Oracle Spatial como plataforma destacada en el sector para la administración de datos espaciales, ofrece gran variedad de características, componentes y herramientas para dar soporte a las necesidades de sistemas de información geográfica. Los siguientes apartados se van a centrar en los componentes y aspectos más básicos e importantes.

# 3.2 Añadir información espacial en las tablas

Considérese una aplicación de negocio que mantiene información de las distintas entidades que forman parte del negocio, tales como clientes, almacenes, proveedores, competidores, etc.

Para obtener los beneficios que puede aportar la explotación de información, tal como la identificación del número de clientes en un determinado territorio, las rutas entre almacenes y clientes, etc., primero es necesario saber cómo incluir esta información espacial en las tablas.

La mayoría de los datos de las aplicaciones pueden ser categorizados en dos conjuntos de tablas:

- Tablas específicas de aplicación: contienen información específica de la aplicación (información sobre clientes, proveedores, almacenes. etc.). Con estas tablas se usan técnica de normalización estándar. Este tipo de datos puede contener información espacial implícita, como direcciones postales.

- Tablas geográficas: son independientes de la aplicación y contiene columnas que almacenan información espacial explícita tal como (los límites de una ciudad o las líneas que sigue una carretera etc.)

En la figura "*Figura 8. Separación de la información*" se pude observar un ejemplo de la separación de esta información para una aplicación de negocio:

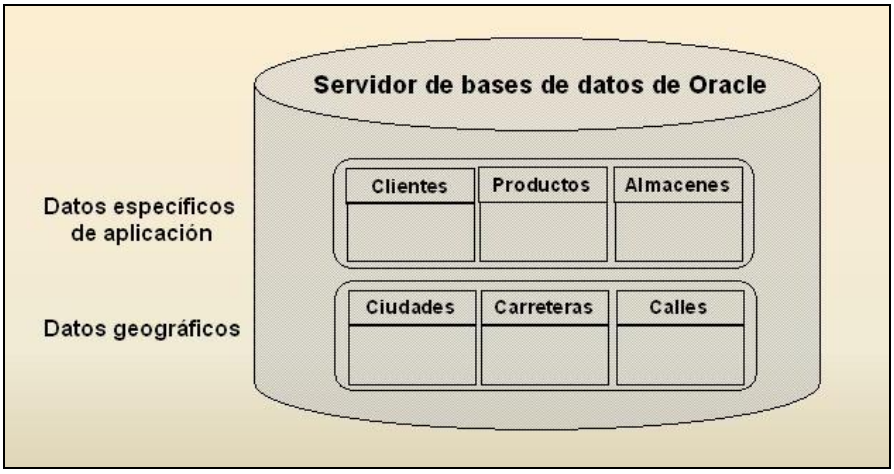



*Figura 8. Separación de la información (Elaboración propia)*

A los datos específicos de la aplicación, también habrá que añadirle información espacial, es decir, habrá que añadir una columna para almacenar información sobre la localización espacial de estas entidades. Esta información básica es almacenada en una columna de la tabla de tipo SDO_GEOMETRY.

Este tipo de objeto es el que proporciona Oracle Spatial para guardar la información espacial. Como se verá más adelante cuando se entre en detalle, sobre la carga de datos, hay diferentes formas de incluir la información en el mismo, como por ejemplo, mediante sentencias INSERT usando el constructor del objeto. Pero el más común, es usar herramientas que proporcionan algunos distribuidores de información espacial, para convertir la información implícita almacenada en las columnas de datos específicos de la aplicación, es decir direcciones postales, en información espacial explicita en el objeto SDO_GEOMETRY: Estas herramientas consultan una base de datos interna para determinar los valores de longitud y latitud. Oracle Spatial, provee una herramienta para realizar el proceso, convirtiendo una dirección postal en un punto bidimensional de la superficie terrestre. Tal vez se necesite realizar un análisis más sofisticado de los datos, en el que la información de un punto bidimensional no sea suficiente, y por lo tanto necesitemos obtener geometrías más complejas. Estos datos están normalmente disponibles desde Distribuidores GIS y agencias de mapeo geográfico, como. NAVTEQ y Tele Atlas, ambas venden datos geográficos sobre Estados Unidos y Europa.

# 3.3 Consideraciones de diseño para datos geográficos

Como ya se ha mencionado anteriormente, para los datos específicos de la aplicación se usan técnicas de diseño estándar como la normalización o el modelo relacional entre otros [KGE, 2007].

Spatial no sigue ningún modelo específico para realizar el mejor diseño de los datos. Establece una estructura jerárquica que se compone de elementos, geometrías y capas:


*Figura 9. Jerarquía de elementos de diseño de datos geográficos (Elaboración propia)*

Las geometrías se ven reflejadas en objetos de tipo SDO_GEOMETRY, y las capas se corresponden con el conjunto de todos los objetos de tipo SDO_GEOMETRY, para una determinada tabla.

Aunque no haya un modelo específico a seguir, a continuación se presentan una serie de criterios generales a establecer, para modelar este tipo de datos:

- Separación de datos que no presenta los mismos atributos, similar a las técnicas de normalización usados con datos normales.

- Separación de datos que se podrían modelar con los mismos atributos, pero que supondría almacenar en una misma tablas un gran volumen de información, que podría perjudicar la consulta de los mismos. Por ejemplo, carreteras y calles pueden presentar las mismas características. Sin embargo el número de calles de un país es mucho mayor que el número de carreteras. Si se juntasen los datos en una misma tabla, la consulta a las carreteras se vería perjudicada en rendimiento.

- Separación basada en la forma geométrica, si es un punto, una línea, un polígono. Como se verá más adelante, esto afecta directamente a la indexación espacial. El rendimiento de un tipo de índice, varía según la geometría a la que se esté aplicando.

- Partición de datos: si el número de registros almacenados en una tabla es muy elevado, la aplicación puede mejorar sus tiempos de respuesta, beneficiándose de realizar particiones en una tabla, para mejorar el tratamiento de los datos.

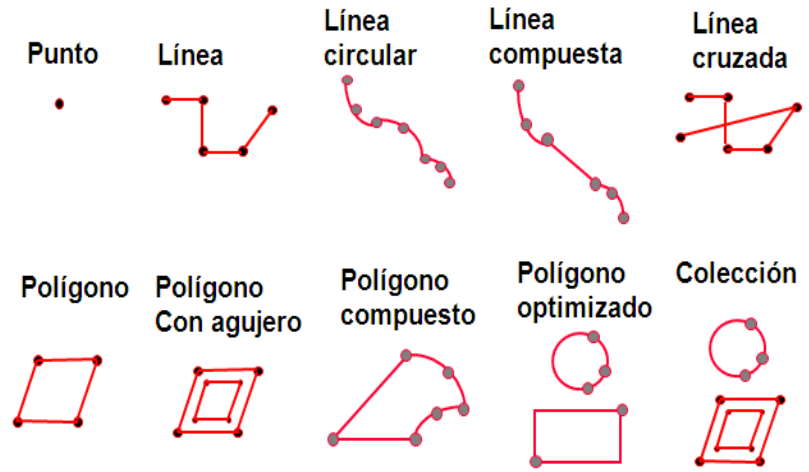
# 3.4 Tipo SDO_GEOMETRY

En el apartado anterior se ha visto como organizar los datos en las tablas en columnas de tipo SDO_GEOMETRY. Este apartado se centra en la modelación y almacenamiento de diferentes tipos de información espacial.

Se va a presentar el tipo de objeto SDO_GEOMETRY, sus atributos y cómo se crean las geometrías haciendo uso del constructor del objeto.

## 3.4.1 Geometrías representadas por el tipo SDO_GEOMETRY

En primer lugar se van a ver los tipos de datos espaciales que se pueden almacenar en el objeto SDO_GEOMETRY. La tabla "*Tabla 1 Geometrías representadas por el objeto SDO_GEOMETRY*" los muestra clasificados en base a los tipos soportados en dos y tres dimensiones.

<table>
<tr>
<td>2D y 3D<br>(Véase *)</td>
<td> 

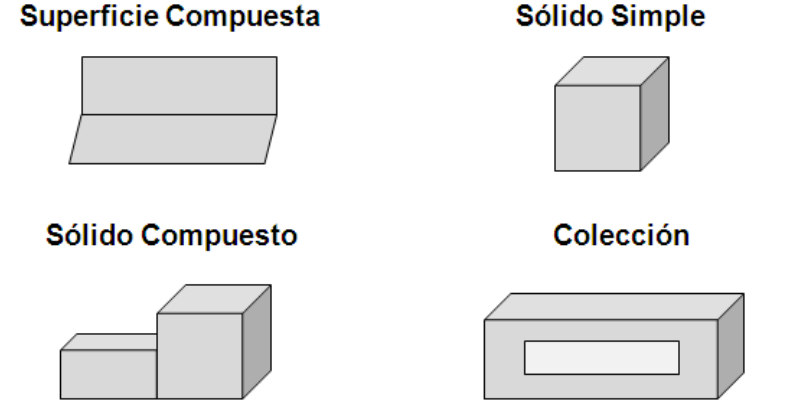
(*) Las geometrías compuestas por arcos no pueden ser representadas en tres dimensiones, Oracle Spatial no soporta arcos ni curvas parametrizadas en geometrías en tres dimensiones.</td>
</tr>
<tr>
<td>Sólo 3D</td>
<td> 
</td>
</tr>
</table>

*Tabla 1 Geometrías representadas por el objeto SDO_GEOMETRY*

Las versiones anteriores de Oracle Spatial 11g, permitían el uso de la tercera y cuarta dimensión para almacenar datos de utilidad asociados a las geometrías, pero realmente

los valores no eran interpretados por Oracle. La versión 11g da soporte completo a objetos 3D, permitiendo construir las geometrías anteriores en tres dimensiones a excepción de aquellas que incluyan arcos o curvas parametrizadas que en la actualidad no son soportadas por Spatial en tres dimensiones.

Más adelante se incluirán ejemplos específicos de geometrías, por el momento se va a ver una breve descripción de los diferentes tipos de geometrías presentados en la tabla "*Tabla 1 Geometrías representadas por el objeto SDO_GEOMETRY*".

- Puntos: es el tipo de geometría más simple, pueden representar las localizaciones de entidades tales como, un cliente, el almacén de un competidor etc.

- Líneas: conecta múltiples puntos o vértices. Si la línea se cierra constituye un anillo. Al menos debe estar formada por dos puntos. Se pueden distinguir diferentes tipos; líneas rectas, formadas por trazos rectos; arcos circulares, formados por trazos circulares; líneas compuestas, combinación de ambos tipos de trazos circulares y rectos. Representan entidades como caminos, rutas de transporte etc.

- Polígonos y Superficies: un polígono está compuesto por uno o más anillos que encierran un área. Puede representar por ejemplo, los límites de una ciudad, un área alrededor de un punto, etc. Se caracterizan por las siguientes propiedades:

    - El límite de un polígono está definido por uno o más anillos (líneas cerradas).

    - Un polígono a diferencia de una línea cerrada, tiene asociada un área encerrada por sus bordes. El área debe ser contigua, es decir, los bordes del polígono no se pueden cruzar. Esto significa que el dígito "8" por ejemplo, no puede ser representado por un único polígono sino como una colección de geometrías.

    - Las líneas que representan los bordes del polígono, pueden ser rectas, curvas o la combinación de ambos, en cuyo caso se denomina polígono compuesto.

    - El área de un polígono puede estar definida mediante un anillo exterior, y un número cualquiera de anillos interiores. El área del polígono exterior, será el resultado de restarle al área encerrada por el polígono exterior las encerradas por los polígonos interiores.

    - Para datos en tres dimensiones, los polígonos están en planos tridimensionales es por lo que se les conoce con el nombre de superficies poligonales. Una restricción a la hora de construir una superficie es que todos los vértices de una superficie poligonal tienen que estar en un mismo plano. Se puede crear una composición de superficies poligonales a partir de superficies poligonales pertenecientes a planos diferentes pero situadas de forma contigua. Un ejemplo de superficies poligonal podría ser el exterior de un edificio.

- Sólidos: Un sólido simple está compuesto por una superficie exterior y cero o más composiciones de superficies interiores. En conjunto la superficie exterior y las composiciones de superficies interiores definen los bordes o límites del sólido. A diferencia de una superficie, un sólido simple tiene tanto un área como un volumen. Se encuentran ejemplos de uso de los sólidos en el modelado de edificios y otros elementos arquitectónicos. En algunos casos dada la complejidad de la estructura del edificio no podrá modelarse haciendo uso de un único sólido, para ello se usarán los sólidos compuestos, consistentes en múltiples sólidos simples con un único volumen.

- Colecciones: Una colección está formada por múltiples elementos geográficos del mismo tipo, colección homogénea, o de distintos tipos, colección heterogénea. Tipos específicos de colecciones homogéneas son los multipuntos, las multilíneas, los multipolígonos, las multisuperficies y los multisólidos.

## 3.4.2 Estructura lógica de SDO_GEOMETRY

Tras presentar los tipos de datos espaciales a manejar, se va a ver la estructura del objeto utilizado para representarlos.

En general, se requiere un número de puntos muy elevados, para definir las figuras espaciales. Por ejemplo, para ilustar, se puede pensar en los límites de un polígono que represente una Comunidad Autónoma. Por ello, SDO_GEOMETRY es implementado usando un array de elementos.

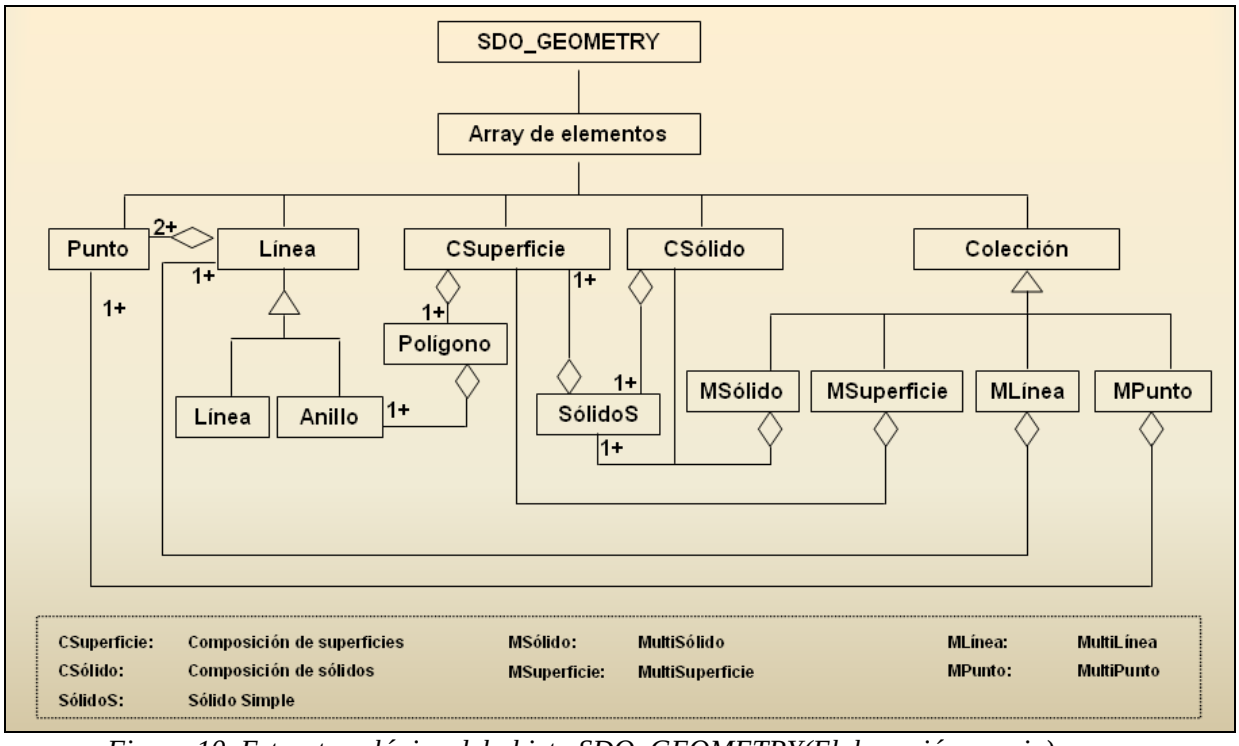



*Figura 10. Estructura lógica del objeto SDO_GEOMETRY(Elaboración propia)*

SDO_GEOMETRY tiene dos componentes lógicos, el sistema de coordenadas de la geometría y el array de elementos que la forma. Este último describe la localización y la forma de la misma.

A continuación se van a ver los atributos en los que se traduce la estructura lógica de la figura "*Figura 10. Estructura lógica del objeto SDO_GEOMETRY*".

## 3.4.3 Atributos del tipo SDO_GEOMETRY

En la figura "*Figura 11. Atributos objeto SDO_GEOMETRY*"se pueden observar los atributos que componen el tipo de objeto SDO_GEOMETRY:

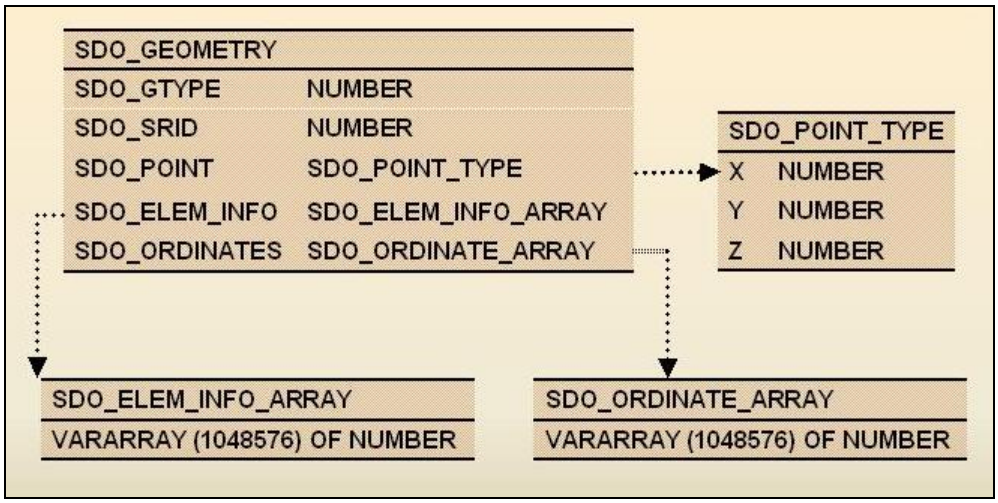



*Figura 11. Atributos objeto SDO_GEOMETRY (Elaboración propia)*

### 3.4.3.1 SDO_GTYPE

Especifica el tipo de figura geométrica que modela el objeto. Como se ha visto anteriormente, la geometría puede estar formada por más de una figura, este atributo se refiere al tipo general que representa el objeto entero.

De forma general tiene la siguiente forma 'D00T'. El primer y último digito 'D' y 'T', son los que marcan el número de dimensiones en las que está expresada la geometría y tipo geometría respectivamente.

| Dígito | Valores |
|---|---|
| D (dimensión de la geometría) | 2 (2 dimensiones)<br>3 (3 dimensiones)<br>4 (4 dimensiones) |
| T (tipo de geometría)<br>Para las geometrías simples | 00 (Tipo no interpretado)<br>01 (Punto)<br>02 (Línea)<br>03 (Polígono) |
| Para las geometrías compuestas<br>(T simple + T colección) | 04 (Colección)<br>05 (Multipunto)<br>06 (Multilínea)<br>07 (Multipolígono) |
| Cubos a Troncos 3D | 08 Sólidos<br>09 Multisólido |

*Tabla 2. Forma general del atributo SDO_GTYPE*

En algunas aplicaciones la tercera y cuarta dimensión, mantiene información adicional para cada uno de los vértices, pero no pertenece a la definición de la forma de la geometría, sino que almacena el valor de una medida determinada. Por ejemplo, en aplicaciones dedicadas a sistemas de transporte, la tercera ordenada de cada vértice en un segmento de un camino almacena el kilómetro en el que se encuentra el punto. Esto es lo que se conoce como sistemas de referencia lineal (LRS). Para geometrías LRS la forma del atributo se adapta a 'DL0T', donde 'D' y 'T' siguen correspondiendo al número de dimensiones y el tipo de geometría respectivamente y 'L' sirve para indicarle a Oracle que dimensión almacena el valor de la medida (3 o 4).

### 3.4.3.2 SDO_SRID

Especifica el sistema de coordenadas de la geometría. Los tipos de sistemas de coordenadas que soporta Spatial son los siguientes:

- Sistema de coordenadas geodésico: si se modela la superficie de la Tierra como un elipsoide tridimensional, se pueden medir las relaciones de distancia entre los objetos, computando la distancia de sus localizaciones correspondientes en el elipsoide. Pero como ya es sabido, la Tierra no es un elipsoide perfecto, por lo que un único elipsoide no puede representar con precisión todas las áreas de la superficie terrestre. Esto llevó a los geógrafos a definir múltiples elipsoides, para cubrir sus necesidades. Oracle Spatial proporciona los más comúnmente usados en la tabla MDSYS.SDO_ELLIPSOIDS. En ocasiones es necesario desplazar el centro de la Tierra y girar los ejes, para adaptarse mejor a la curvatura de una determinada región. Para ello se crean modelos de desplazamiento y rotación de elipsoides concretos para adaptarse mejor a la curvatura terrestre de diferentes regiones. Estos modelos se conocen como datums. Se pueden examinar los diferentes modelos tridimensionales de datums en la tabla MDSYS.SDO_DATUMS. Al posicionamiento de datos sobre la superficie de la Tierra al referirse a las coordenadas en un datum específico, es a lo que se le denomina sistema de coordenadas geodésico.

- Sistema de coordenadas proyectado: en la mayoría de las aplicaciones, los datos están concentrados en una región específica de la tierra. Proyectar cada dato de la superficie a un plano en dos dimensiones, puede proporcionar una representación más simple y puede ser más preciso para las necesidades de la aplicación. Puesto que no hay una técnica que mantenga todas las características de una región de partida intacta, tales como la distancia, el área o la dirección de los objetos, existen diferentes técnicas que se pueden aplicar, según las características que se deseen mantener. Las diferentes técnicas que se pueden aplicar a un datum, están almacenadas en la tabla MDSYS.SDO_PROJECTIONS.

- Sistema de coordenadas local: los sistemas de coordenadas relativos a la superficie de la tierra, se denominan georeferenciados. El resto de sistemas como los usados en el entono CAD/CAM se denominan locales o no georeferenciados.

Se puede establecer el sistema de coordenadas más adecuado para una determinada aplicación teniendo en cuenta las características de la misma.

Si el sistema no es georeferenciado se establece el valor NULL (sistemas de coordenadas cartesianas) o si se toman los datos específicos de un proveedor, el que fije sus datos.

Si por el contrario el sistema es georeferenciado, se puede establecer un sistema de coordenadas geodésico o proyectado. Los sistemas proyectados son idóneos para pequeñas superficies y cuando las necesidades de la aplicación, requieren preservar distancias, formas, áreas o cualquier característica de este tipo. Los sistemas de coordenadas geodésicos son apropiados cuando las superficies con las que se debe trabajar son más grandes, y se puede tolerar algunos errores leves, en las características mencionadas anteriormente.

En la tabla MDSYS.CS_SRS se encuentran los valores que proporciona Oracle para el atributo SDO_SRID:

| Campo | Descripción |
|---|---|
| CS_NAME | Nombre del sistema |
| SRID | identificador numérico del sistema |
| AUTH_SRID, AUTH_NAME | Valores informativos asignados por el originador del sistema |
| WKTEXT | String que contiene información detallada sobre el sistema de coordenadas. Para sistemas geodésicos empieza con el prefijo ‘GEOCS’, para sistemas proyectados con ‘PROJCS’ y para sistemas locales con el prefijo ‘LOCAL_CS’. |

*Tabla 3. Campos tabla CS_SRS*

Realizando la siguiente consulta sobre esta tabla se puede localizar por ejemplo, un sistema de coordenadas proyectado para Europa:

```
SELECT *
FROM mdsys.cs_srs
WHERE cs_name LIKE '%Europe%'
AND wktext LIKE '%PROJCS%'
```

*Figura 12. Ejemplo de selección de sistemas proyectado*

Se obtendrían dos sistemas proyectados:

- Equal-Area Projection (Europe)
- Conformal Projection (Europe)

Oracle soporta gran cantidad de sistemas de coordenadas que cubren la mayoría de las regiones del mundo. La tabla CS_SRS permite obtener toda la información relevante sobre sistemas de coordenadas 2D. Sin embargo, la información sobre sistemas de coordenadas 1D y 3D registrada en esta tabla es parcial, por lo que para obtener la información completa también se debe hacer uso de las tablas como SDO_COORD_REF_SYS y SDO_CRS_VERTICAL.

Es posible que esto cambie en revisiones posteriores de la versión 11g de modo que con consultar la tabla CS_SRS sea suficiente.

### 3.4.3.3 SDO_POINT

Se usa cuando la geometría que se va a representar es un punto. Es de tipo SDO_POINT_TYPE. Para dar valor a este atributo, se debe usar su constructor, como con cualquier otro tipo de objeto. Véase la siguiente sentencia:

```
INSERT INTO geometrias (nombre, descripcion, geometria) VALUES
        ('Punto',
         'Punto en dos dimensiones',
         SDO_GEOMETRY
         (2001,                          -- SDO_GTYPE: Punto en dos dimensiones
          8225,                          -- SDO_SRID: Sistema geodésico
          SDO_POINT_TYPE
                (-85,                    -- Ordenada para el valor de la longitud
                  37,                    -- Ordenada para el valor de la latitud
                  NULL),                 -- No hay ordenada para la 3 ª dimensión
         NULL,
         NULL)
```

*Figura 13. Ejemplo de selección de sistemas proyectado*

Si se está representando un punto en un sistema de coordenadas geodésico, Oracle obliga a que la primera coordenada sea siempre la longitud, y la segunda la latitud.

SDO_POINT solo puede recibir tres ordenadas (x, y, z) para datos de más de tres dimensiones habrá que usar los atributos SDO_ELEM_INFO, SDO_ORDINATES.

### 3.4.3.4 SDO_ORDINATES

Almacena en orden las coordenadas de los vértices de todos los elementos que componen la geometría. Como se puede observar en la figura "*Figura 11. Atributos objeto SDO_GEOMETRY*" es de tipo SDO_ORDINATE_ARRAY, que es una colección de tipo VARRAY de números. Si la geometría está representada en D dimensiones, todos los D números consecutivos en SDO_ORDINATES, especifican las coordenadas de un vértice. Por ejemplo si se quiere modelar una línea en dos dimensiones compuesta por dos puntos A ($X_A$, $Y_A$) y B ($X_B$, $Y_B$), el array contendrá los números en este orden ($X_A$, $Y_A$, $X_B$, $Y_B$). Se puede observar que por sí solo este objeto contiene una colección de números que por sí solos, no dan más información. Cabe preguntarse, cómo son separadas e interpretadas estas coordenadas. Esta información es especificada en el atributo SDO_ELEM_INFO.

### 3.4.3.5 SDO_ELEM_INFO

El atributo SDO_ELEM_INFO, es de tipo SDO_ELEM_INFO_ARRAY, implementado también mediante un array de números, agrupados todos ellos de tres en tres números consecutivos. Cada triplete describe un elemento de la geometría y es de la forma <<desplazamiento, tipo de elemento, interpretación>>. El desplazamiento indica el índice de la primera ordenada del elemento en el array. Los números correspondientes al tipo de elemento e interpretación, toman diferentes valores dependiendo del tipo de elemento que representen punto, línea o polígono y de cómo estén conectados los bordes de la misma, mediante trazos rectos, curvos o ambos. La tabla "*Tabla 4 Desplazamiento, Tipo de Elemento, Interpretación*" resume los valores posibles:

| Nombre | Tipo | Interpretación | |
|---|---|---|---|
| **Punto** | **1** | **N** | Número de puntos de la colección. 1 para un punto simple. |
| **Línea** | **2** | **1** | Vértices conectados por líneas rectas. |
| | | **2** | Vértices conectados por arcos. |
| **Polígono Exterior** | **1003** | **1** | Vértices conectados por líneas rectas. |
| | | **2** | Vértices conectados por arcos. |
| | | **3** | Rectángulo optimizado, definido solo por dos vértices, inferior izquierdo y superior derecho. |
| | | **4** | Circunferencia, definida por tres vértices. |
| **Polígono Interior** | **2003** | **1** | Vértices conectados por líneas rectas. |
| | | **2** | Vértices conectados por arcos. |
| | | **3** | Rectángulo optimizado, definido solo por dos vértices, inferior izquierdo y superior derecho. |
| | | **4** | Circunferencia, definida por tres vértices. |
| **Línea Compuesta** | **4** | **N** | Número de cambios de tipos de subelemento que constituyen la línea compuesta. Este valor se especifica en la línea de cabecera de la geometría, a esta le siguen las tuplas de los elementos que componen la misma y en las que la interpretación toma los valores 1 y 2 dependiendo de sí el subelemento corresponde a un trazo recto o curvo. |

| Nombre | Tipo | Interpretación | |
|---|---|---|---|
| **Polígono Compuesto Exterior** | **1005** | **N** | Número de cambios de tipos de subelemento que constituyen el polígono compuesto. Este valor se especifica en la línea de cabecera de la geometría, a esta le siguen las tuplas de los elementos que componen la misma y en las que la interpretación toma los valores 1 y 2 dependiendo de si el subelemento corresponde a un trazo recto o curvo o 3 si es a su vez se trata de la línea de cabecera de polígono interior al mismo |
| **Polígono Compuesto Interior** | **2005** | **N** | Número de cambios de tipos de subelemento que constituyen el polígono compuesto. Este valor se especifica en la línea de cabecera de la geometría, a esta le siguen las tuplas de los elementos que componen la misma y en las que la interpretación toma los valores 1 y 2 dependiendo de si el subelemento corresponde a un trazo recto o curvo o 3 si es a su vez se trata de la línea de cabecera de polígono interior al mismo |

*Tabla 4 Desplazamiento, Tipo de Elemento, Interpretación*

Es importante recordar que los arcos y curvas parametrizadas no son soportados por Oracle Spatial en tres dimensiones, por lo que los valores de interpretación que representen arcos y los tipos correspondientes a geometrías compuestas de trazos rectos y arcos, no pueden utilizarse cuando se estén definiendo geometrías en tres dimensiones.

### 3.4.4 Creación de tablas con datos espaciales

Una vez definido el tipo SDO_GEOMETRY y sus atributos, cabe preguntarse cómo crear las tablas en las que insertar las geometrías. En principio sólo se necesita añadir una columna a la tabla de tipo SDO_GEOMETRY.

La siguiente figura muestra la creación de las tablas 'PUNTOS', 'LINEAS', 'POLIGONOS' y 'GEO3D', que se utilizarán como segundo escenario para los ejemplos de manejo de datos espaciales en los siguientes apartados:

```
CREATE TABLE PUNTOS(
   CD_PUNTO NUMBER NOT NULL,
   DESCRIPCION VARCHAR2(32),
   GEOMETRIA MDSYS.SDO_GEOMETRY);

CREATE TABLE LINEAS(
   CD_LINEA NUMBER NOT NULL,
   DESCRIPCION VARCHAR2(32),
   GEOMETRIA MDSYS.SDO_GEOMETRY);

CREATE TABLE POLIGONOS(
   CD_POLIGONO NUMBER NOT NULL,
   DESCRIPCION VARCHAR2(32),
   GEOMETRIA MDSYS.SDO_GEOMETRY);

CREATE TABLE GEO3D(
   CD_GEO3D NUMBER NOT NULL,
   DESCRIPCION VARCHAR2(32),
   GEOMETRIA MDSYS.SDO_GEOMETRY);
```

*Figura 14. Tablas de ejemplo de creación de geometrías*

Como se puede observar las tablas mantienen columnas correspondientes a información no espacial, como 'CD_PUNTO' y 'DESCRIPCIÓN'; y una columna 'GEOMETRIA' del tipo SDO_GEOMETRY donde insertar las geometrías.

La definición de la columna de tipo SDO_GEOMETRY sigue la misma sintaxis que cualquier otro tipo de columna. Como ocurre también con cualquier otro campo, se pueden crear índices sobre la columna para optimizar las consultas.

Además de crear la columna de tipo SDO_GEOMETRY es necesario indicar lo que se conoce con el nombre de metadatos correspondientes a la tabla o capa, esta información se mantiene en una vista propia de Oracle Spatial, USER_SDO_GEOM_METADATA.

Los metadatos a indicar para cada una de las tablas y la creación de índices espaciales se verán con detalle más adelante, vamos a ver primero varios ejemplos que nos ayuden a comprender mejor los valores a indicar en los atributos del objeto SDO_GEOMETRY para construir las geometrías deseadas.

## 3.4.5 Ejemplos de geometrías simples

Una geometría simple es un punto, un polígono o una línea que queda perfectamente determina, usando un único triplete. Es decir el atributo SDO_ELEM_INFO tiene siempre el siguiente formato (1, x, y). Los valores que tomará x e y dependerán como ya se ha mencionado anteriormente, del tipo de elemento que forma la geometría y de cómo estén conectados los vértices de la misma.

A continuación se puede observar la representación de algunos ejemplos de geometrías simples y las sentencias SQL correspondientes.

### 3.4.5.1 Punto Bidimensional

La siguiente sentencia SQL insertaría un punto bidimensional. En este caso no es necesario usar la definición el atributo SDO_ELEM_INFO. De hecho aunque es posible, no es recomendable. La mejor opción es usar el atributo personalizado para un punto SDO_POINT.

```
INSERT INTO puntos VALUES
(
        1 ,    --  código de la geometría, información no espacial
        'Punto1', -- descripción de la geometría, información no espacial
SDO_GEOMETRY
        (
                2001, -- SDO_GTYPE.  Punto en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                SDO_POINT
                (
                        2,              -- ordenada 1
                        56,             -- ordenada 2
                        NULL,  -- sólo dos dimensiones
                ),
                NULL, -- Como es un punto no se usa SDO_ELEM_INFO_ARRAY
                NULL  -- Como es un punto no se usa SDO_ORDINATE_ARRAY
        )
)
```

*Figura 15. Ejemplo de punto en dos dimensiones*

### 3.4.5.2 Línea Simple bidimensional conectada por trazos rectos

En la siguiente figura se puede observar la representación correspondiente a una línea conectada por trazos rectos, en un sistema bidimensional.

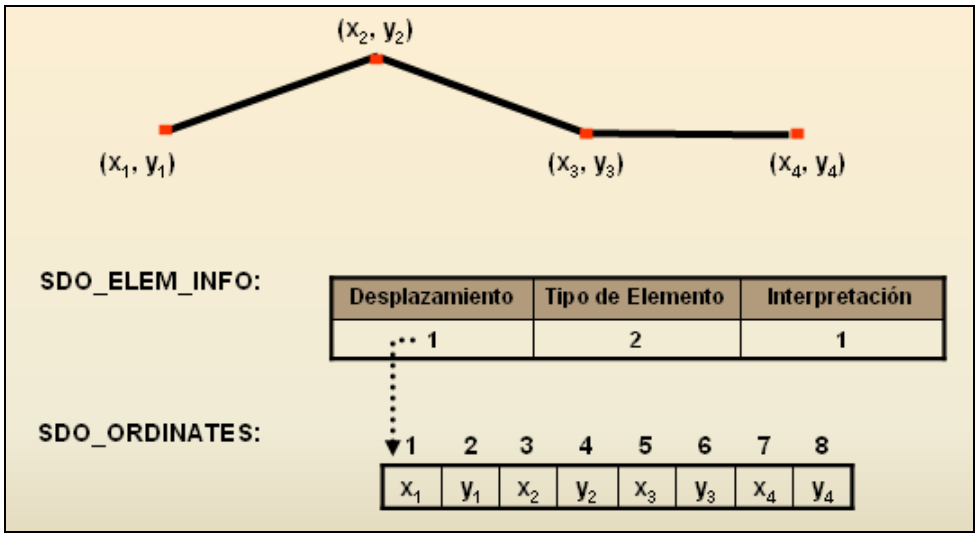



*Figura 16. Ejemplo línea conectada por trazos rectos (Elaboración propia)*

La sentencia SQL para añadir esta línea con el código '1' y la descripción 'Linea1' en la tabla 'LINEAS' sería la siguiente:

```
INSERT INTO lineas VALUES
(       1 ,    --  código de la geometría, información no espacial
        'Línea 1', -- descripción de la geometría, información no espacial
        SDO_GEOMETRY
        (       2002,    -- SDO_GTYPE.  Línea en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,          -- Las ordenadas del único elemento que componen la
                                    -- geometría comienzan en la posición 1 del array
                        2,          -- El tipo de elemento es 2 por ser una Línea
                                    -- La interpretación es 1 porque sus vértices están
                                    -- conectados por trazos rectos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1, -- Valores ordenadas punto 1
                        X2, Y2, -- Valores ordenadas punto 2
                        X3, Y3, -- Valores ordenadas punto 3
                        X4, Y4 -- Valores ordenadas punto 4
                )
        )
)
```

*Figura 17. Ejemplo sentencia de línea conectada por trazos rectos*

### 3.4.5.3 Línea simple bidimensional conectada por trazos curvos

En la siguiente figura se puede observar la representación correspondiente a una línea conectada por trazos curvos, en un sistema bidimensional.

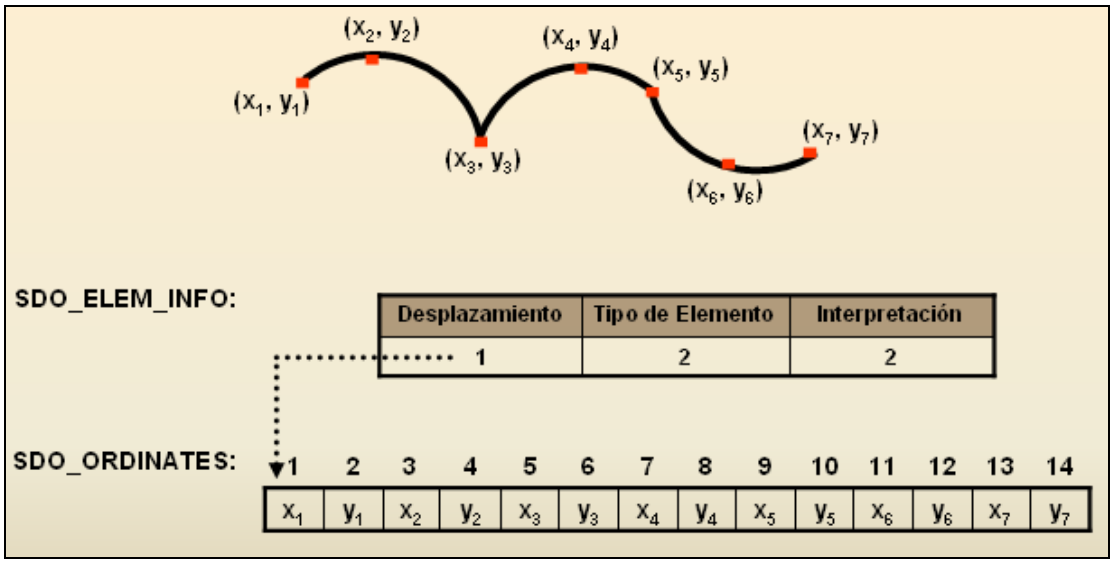



*Figura 18. Ejemplo de línea conectada por trazos curvos (Elaboración propia)*

La sentencia SQL para añadir esta línea con el código '2' y la descripción 'Linea2' en la tabla 'LINEAS' sería la siguiente:

```
INSERT INTO lineas VALUES
(       2,    --  código de la geometría, información no espacial
        'Línea 2', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2002,    -- SDO_GTYPE.  Línea en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Las ordenadas del único elemento que componen la
                                  -- geometría comienzan en la posición 1 del array
                        2,        -- El tipo de elemento es 2 por ser una Línea
                                  -- La interpretación es 2 porque sus vértices están
                                  -- conectados por arcos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,  -- Valores ordenadas punto 1
                        X2, Y2,  -- Valores ordenadas punto 2
                        X3, Y3, -- Valores ordenadas punto 3
                        X4, Y4,  -- Valores ordenadas punto 4
                        X5, Y5, -- Valores ordenadas punto 5
                        X6, Y6,-- Valores ordenadas punto 6
                        X7, Y7 -- Valores ordenadas punto 7
                )
        )
)
```

*Figura 19. Ejemplo sentencia de línea conectada por trazos curvos*

### 3.4.5.4 Polígono simple bidimensional conectado por trazos rectos

En la siguiente figura se puede observar la representación correspondiente a un polígono conectado por trazos rectos, en un sistema bidimensional.

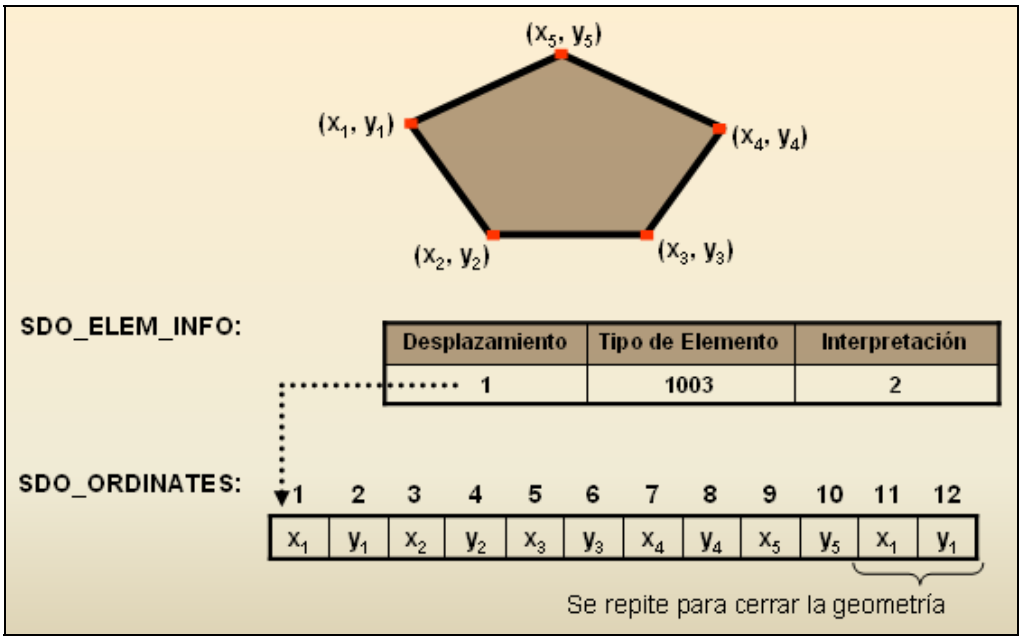



*Figura 20. Ejemplo de polígono conectado por trazos rectos (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código '1' y la descripción 'Poligono1' en la tabla 'POLIGONOS' sería la siguiente:

```
INSERT INTO poligonos VALUES
(       1,    --  código de la geometría, información no espacial
        'Polígono 1', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,          -- Las ordenadas del único elemento que componen la
                                    -- geometría comienzan en la posición 1 del array
                        1003,    -- El tipo de elemento es 1003 por ser un Polígono
                Exterior
                        1          -- La interpretación es 1 porque sus vértices están
                                    -- conectados por trazos rectos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1, -- Valores ordenadas punto 1
                        X2, Y2, -- Valores ordenadas punto 2
                        X3, Y3, -- Valores ordenadas punto 3
                        X4, Y4 , -- Valores ordenadas punto 4
                        X5, Y5, -- Valores ordenadas punto 5
                        X1,Y1 -- Valores ordenadas punto 1 repetido (cierra la geometría)
                )
        )
```

*Figura 21. Ejemplo sentencia de polígono conectado por trazos rectos*

### 3.4.5.5 Polígono simple bidimensional conectado por trazos curvos

En la siguiente figura se puede observar la representación correspondiente a un polígono conectado por trazos curvos, en un sistema bidimensional.

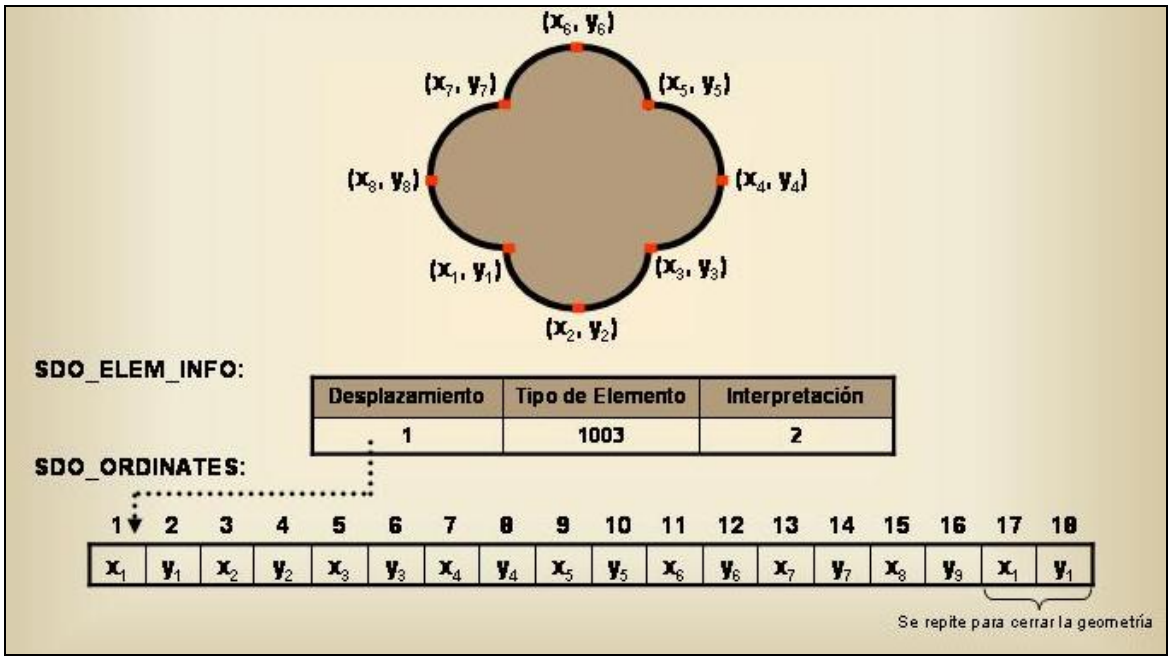



*Figura 22. Ejemplo de polígono conectado por trazos curvos (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código '2' y la descripción 'Polígono2' en la tabla 'POLIGONOS' sería la siguiente:

```
INSERT INTO puntos VALUES
(       2,    --  código de la geometría, información no espacial
        'Polígono 2', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Las ordenadas del único elemento que componen la
                                  -- geometría comienzan en la posición 1 del array
                        1003,   -- El tipo de elemento es 1003 por ser un Polígono
                                  -- Exterior
                        2         -- La interpretación es 2 porque sus vértices están
                                  -- conectados por arcos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,  -- Valores ordenadas punto 1
                        X2, Y2,  -- Valores ordenadas punto 2
                        X3, Y3, -- Valores ordenadas punto 3
                        X4, Y4 , -- Valores ordenadas punto 4
                        X5, Y5,  -- Valores ordenadas punto 5
                        X6, Y6,  -- Valores ordenadas punto 6
                        X7, Y7,  -- Valores ordenadas punto 7
```

```
                        X8, Y8,  -- Valores ordenadas punto 8
                        X1, Y1   -- Valores ordenadas punto 1 repetido para cerrar
                                 -- la geometría
                )
        )
)
```

*Figura 23. Ejemplo sentencia de polígono conectado por trazos curvos*

### 3.4.5.6 Rectángulo optimizado bidimensional

En la siguiente figura se puede observar la representación correspondiente a un rectángulo optimizado, en un sistema bidimensional.

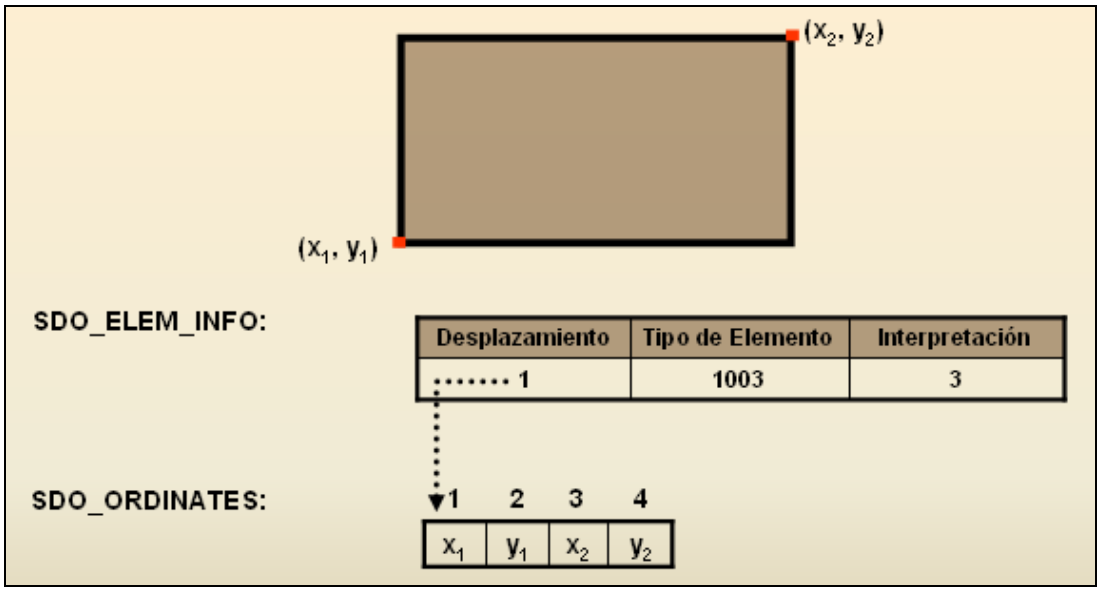



*Figura 24. Ejemplo rectángulo optimizado (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código '3' y la descripción 'Polígono 3' en la tabla 'POLIGONOS' sería la siguiente:

```
INSERT INTO poligonos VALUES
(       3,    --  código de la geometría, información no espacial
        'Poligono 3', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Las ordenadas del único elemento que componen la
                                  -- geometría comienzan en la posición 1 del array
                        1003,   -- El tipo de elemento es 1003 por ser un Polígono
                Exterior
                        3         -- La interpretación es 3 porque se trata de un rectángulo
                                  -- optimizado, definido por dos vértices, inferior
                                  -- izquierdo y superior derecho
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1, -- Valores ordenadas punto 1
                        X2, Y2 -- Valores ordenadas punto 2
                )
        )
```

*Figura 25. Ejemplo sentencia rectángulo optimizado*

### 3.4.5.7 Circunferencia optimizada bidimensional

En la siguiente figura se puede observar la representación correspondiente a una circunferencia optimizada, en un sistema bidimensional.

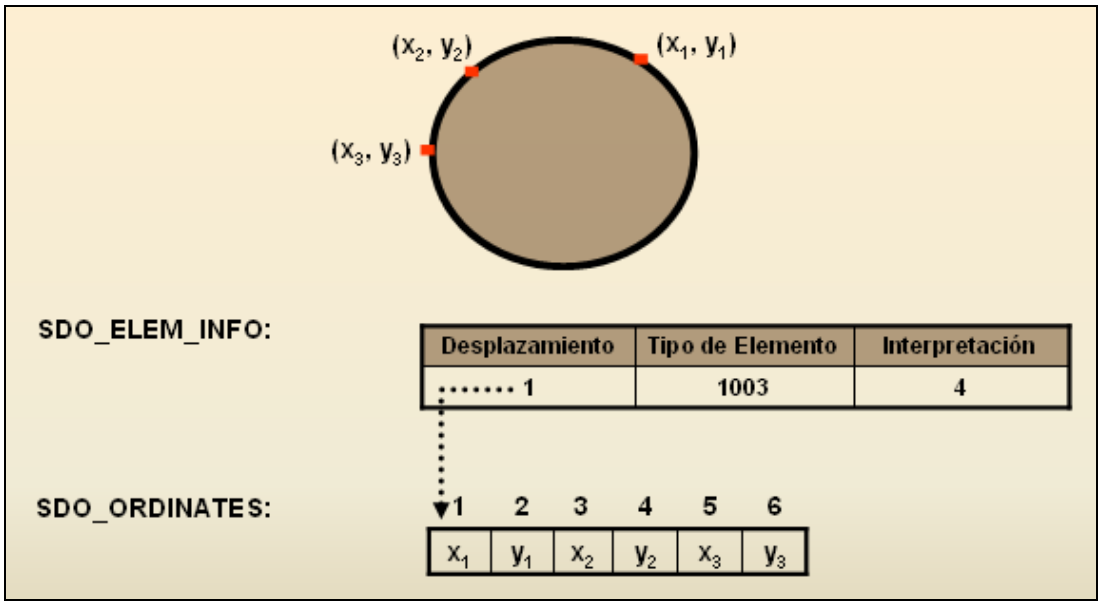



*Figura 26. Ejemplo de circunferencia optimizada (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código '4' y la descripción 'Polígono 4' en la tabla 'POLIGONOS' sería la siguiente:

```
INSERT INTO poligonos VALUES
(       4,    --  código de la geometría, información no espacial
        'Poligono 4', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Las ordenadas del único elemento que componen la
                                  -- geometría comienzan en la posición 1 del array
                        1003,   -- El tipo de elemento es 1003 por ser un Polígono
                Exterior
                        4         -- La interpretación es 4 porque se trata de una
                                  -- circunferencia, definida por tres vértices
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,  -- Valores ordenadas punto 1
                        X2, Y2,  -- Valores ordenadas punto 2
                        X3, Y3, -- Valores ordenadas punto 3
                )
        )
```

*Figura 27. Ejemplo sentencia de circunferencia optimizada*

## 3.4.6 Ejemplos de geometrías compuestas

A continuación se puede observar la representación de algunos ejemplos de geometrías compuestas y las sentencias SQL correspondientes.

### 3.4.6.1 Línea compuesta bidimensional

En la siguiente figura se puede observar la representación correspondiente a una línea compuesta, en un sistema bidimensional:

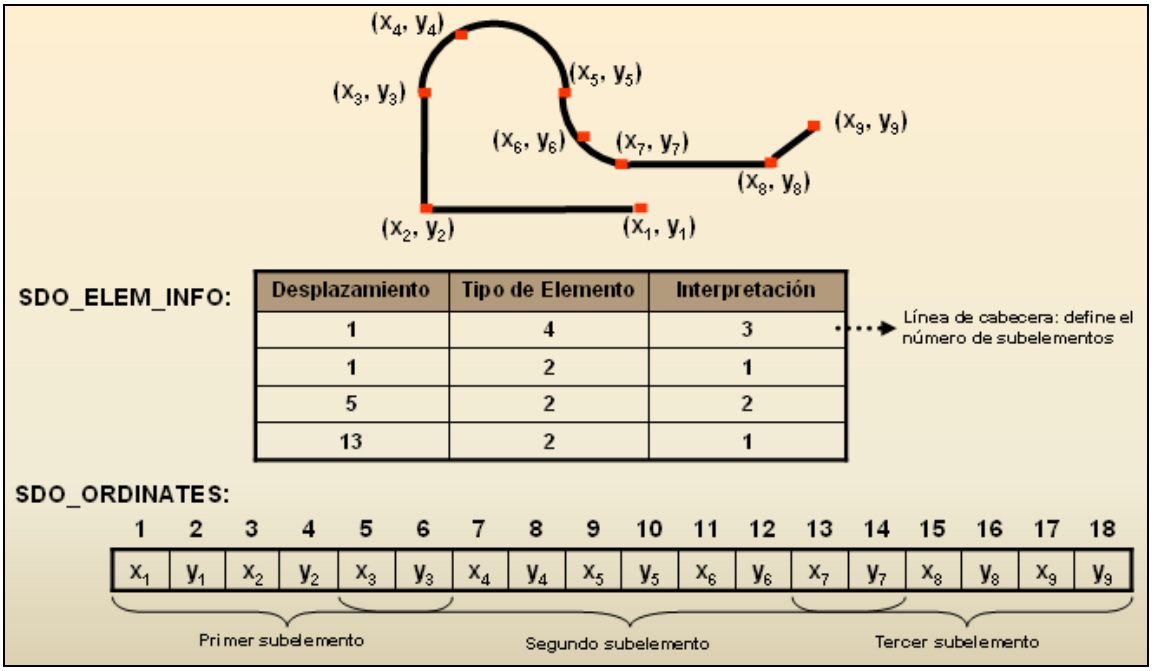



*Figura 28. Ejemplo de línea compuesta (Elaboración propia)*

La sentencia SQL para añadir esta línea con el código '3' y la descripción 'Línea 3' en la tabla 'LINEAS' sería la siguiente:

```
INSERT INTO lineas VALUES
(       3,    --  código de la geometría, información no espacial
        'Línea 3', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2002,    -- SDO_GTYPE.  Línea en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Línea de cabecera, por definición es 1
                        4,        -- El tipo de elemento es 4 por ser una Línea compuesta
                        3,        -- La interpretación es 3 porque es el número de cambios
                                  -- de tipos de subelementos que constituyen la línea –
                                  -- compuesta. Este valor se especifica en la línea de –
                                  -- cabecera de la geometría, a esta le siguen las tuplas
                                  de
                                  -- los elementos que componen la misma y en las que la
                                  -- interpretación toma los valores 1 y 2 dependiendo de si
```

```
                        1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1,      -- La interpretación es 1 porque sus vértices están
                                -- conectados por trazos rectos
                        5,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 5 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        2,      -- La interpretación es 2 porque sus vértices están
                                -- conectados por arcos
                        13,     -- La ordenadas del tercer elemento que componen la
                                -- geometría comienzan en la posición 13 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1       -- La interpretación es 1 porque sus vértices están
                                -- conectados por trazos rectos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,   -- Valores ordenadas punto 1 (Inicio primer subelemento)
                        X2, Y2,   -- Valores ordenadas punto 2
                        X3, Y3,   -- Valores ordenadas punto 3 (Fin primer subelemento Y3,
                                  -- inicio segundo subelemento X3)
                        X4, Y4,   -- Valores ordenadas punto 4
                        X5, Y5,   -- Valores ordenadas punto 5
                        X6, Y6,   -- Valores ordenadas punto 6
                        X7, Y7,   -- Valores ordenadas punto 7 (Fin segundo subelemento
                                  --Y7, inicio tercer subelemento X7)
                        X8, Y8, -- Valores ordenadas punto 8
                        X9 Y9   -- Valores ordenadas punto 9 (Fin tercer subelemento Y9)
                )
        )
)
```

*Figura 29. Ejemplo sentencia de línea compuesta*

### 3.4.6.2 Polígono compuesto bidimensional

En la siguiente figura se puede observar la representación correspondiente a un polígono compuesto, en un sistema bidimensional:

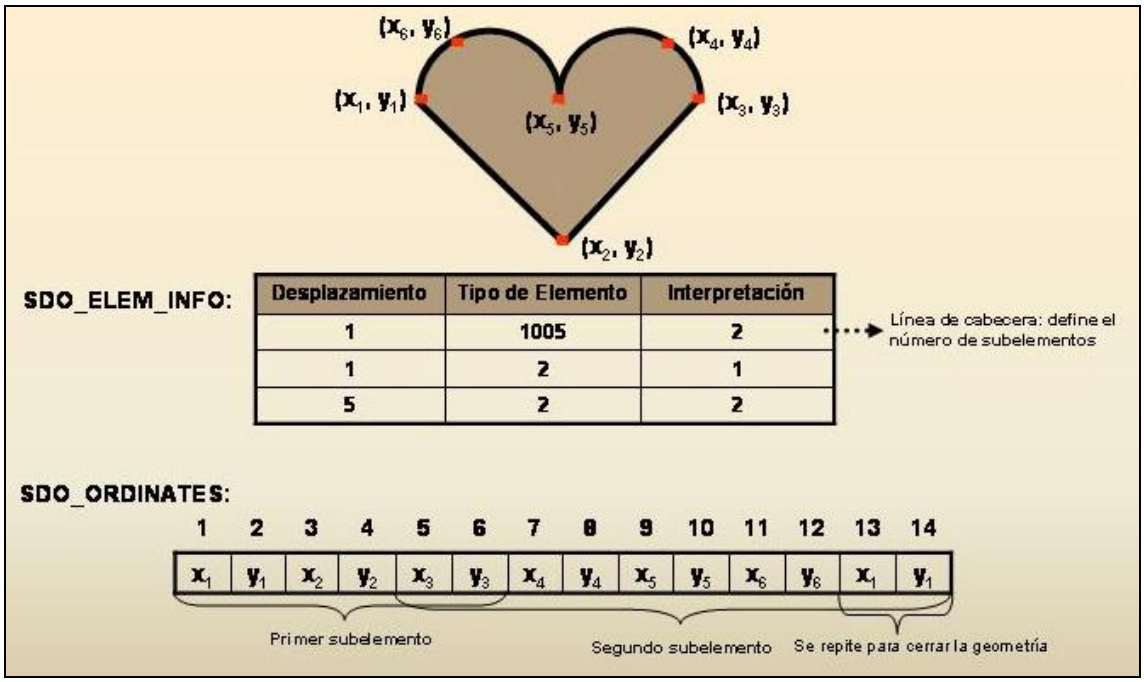



*Figura 30. Ejemplo de polígono compuesto (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código '5' y la descripción 'Polígono 5' en la tabla 'POLIGONOS' sería la siguiente:

```
INSERT INTO poligonos VALUES
        (5,    --  código de la geometría, información no espacial
        'Polígono 5', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,      -- Línea de cabecera, por definición es 1
                        1005,   -- El tipo de elemento es 1005 por ser un Polígono
                                -- compuesto exterior
                        2,      -- La interpretación es 2 porque es el número de cambios
                                -- de tipos de subelementos que constituyen el polígono
                                -- exterior
                        1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1,      -- La interpretación es 1 porque sus vértices están
                                -- conectados por trazos rectos
                        5,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 5 del array
```

```
                    2,      -- El tipo de elemento es 2 por ser una Línea
                    2       -- La interpretación es 2 porque sus vértices están
                            -- conectados por arcos
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,  -- Valores ordenadas punto 1  (Inicio primer subelemento)
                        X2, Y2,  -- Valores ordenadas punto 2
                        X3, Y3,  -- Valores ordenadas punto 3 (Fin primer subelemento Y3,
                                 -- inicio segundo subelemento X3)
                        X4, Y4,  -- Valores ordenadas punto 4
                        X5, Y5,  -- Valores ordenadas punto 5
                        X6, Y6,  -- Valores ordenadas punto 6
                        X1, Y1   -- Valores ordenadas punto 1 (Fin segundo subelemento,
                                 -- se repiten para cerrar la  geometría)
                )
        )
)
```

*Figura 31. Ejemplo sentencia de polígono compuesto*

### 3.4.6.3 Polígono con Agujero

En la siguiente figura se puede observar la representación correspondiente a un polígono con agujero, en un sistema bidimensional:

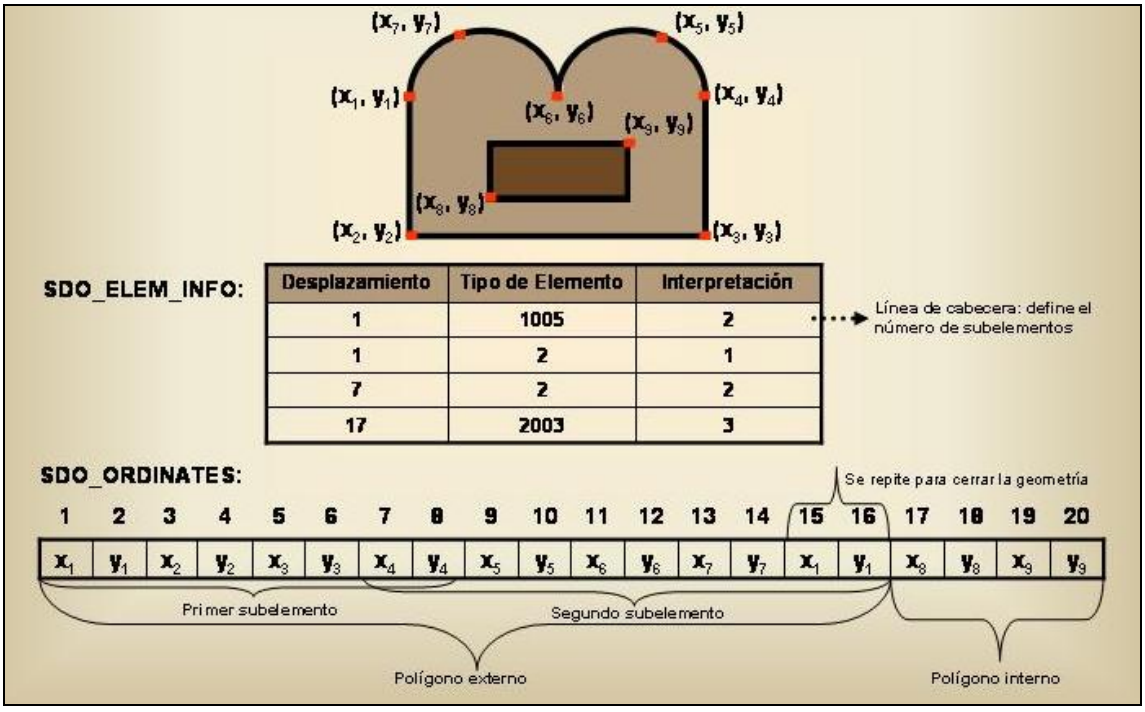



*Figura 32. Ejemplo de polígono con agujero (Elaboración propia)*

La sentencia SQL para añadir este polígono con el código ‘6’ y la descripción ‘Polígono 6’ en la tabla ‘POLIGONOS’ sería la siguiente:

```
INSERT INTO poligonos VALUES
(       6,    --  código de la geometría, información no espacial
        'Polígono 6', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2003,    -- SDO_GTYPE.  Polígono en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,      -- Línea de cabecera, por definición es 1
                        1005,   -- El tipo de elemento es 1005 por ser  un Polígono
                                -- compuesto exterior
                        2,      -- La interpretación es 2 porque es el número de cambios
                                -- de tipos de subelementos que constituyen el polígono
                                -- exterior
                        1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1,      -- La interpretación es 1 porque sus vértices están
                                -- conectados por trazos rectos
                        7,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 7 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        2,      -- La interpretación es 2 porque sus vértices están
                                -- conectados por arcos
                        17,     -- Las ordenadas del elemento que compone el polígono
                                -- interior comienzan en la posición 17 del array
                        2005,   -- El tipo de elemento es 2005 por ser un Polígono
                                -- compuesto interior
                        3       -- La interpretación es 3 porque es el correspondiente a
                                un -- Rectángulo optimizado.
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1,  -- Valores ordenadas punto 1 (Inicio primera
                                 -- geometría, inicio primer subelemento)
                        X2, Y2,  -- Valores ordenadas punto 2
                        X3, Y3,  -- Valores ordenadas punto 3
                        X4, Y4,  -- Valores ordenadas punto 4 (Fin primer subelemento Y4,
                                 -- inicio segundo subelemento X4)
                        X5, Y5,  -- Valores ordenadas punto 5
                        X6, Y6,  -- Valores ordenadas punto 6
                        X7 Y7,   -- Valores ordenadas punto 7
                        X1, Y1,  -- Valores ordenadas punto 1 (Fin segundo subelemento
                                 -- se repiten para cerrar la geometría, fin primera
                                 -- geometría
                        X8, Y8   -- Valores ordenadas punto 8 (Inicio segunda geometría)
                        X9, Y9   -- Valores ordenadas punto 9 (Fin segunda geometría)
                )
        )
)
```

*Figura 33. Ejemplo sentencia de polígono con agujero*

## 3.4.7 Ejemplos de geometrías en tres dimensiones

Todos los ejemplos vistos hasta el momento representan geometrías en un espacio de dos dimensiones. Seguidamente se incluyen algunos ejemplos propios de escenarios tridimensionales.

### 3.4.7.1 Superficie con agujero tridimensional

En la siguiente figura se puede observar la representación correspondiente a una superficie con agujero, en un sistema tridimensional.

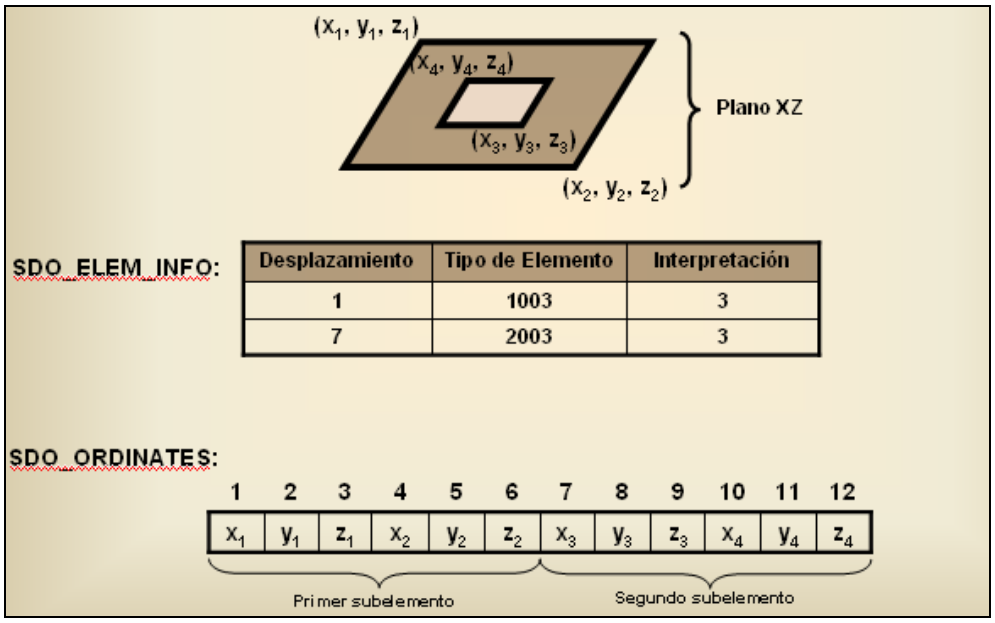



*Figura 34. Ejemplo de superficie con agujero (Elaboración propia)*

En los ejemplos de geometrías en dos dimensiones, se indicó como Oracle establece reglas con respecto al orden de los vértices en función de si el anillo es externo o interno. En geometrías en tres dimensiones la restricción es que el orden de los vértices de los anillos internos y externos sean contrarios, pero no establece cual debe ser el orden para cada uno en concreto. Sin embargo existe una restricción cuando se trata de rectángulos optimizados, si el anillo exterior de define por <esquina mínima - esquina máxima>, el interior debe hacerse al contrario, es decir, <esquina máxima – esquina mínima> (y viceversa). Nótese que esta norma difiere de la establecida para rectángulos en dos dimensiones, donde el rectángulo siempre se definía como <esquina mínima - esquina máxima> independientemente de si el anillo era interno o externo.

La sentencia SQL para añadir esta superficie con agujero con el código ‘1’ y la descripción ‘Superficie 1’ en la tabla ‘GEO3D’ sería la siguiente:

```
INSERT INTO geo3d VALUES
(       1,    --  código de la geometría, información no espacial
        'Superficie 1', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       3003,    -- SDO_GTYPE. Superficie en tres dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array.
                        1003,   -- El tipo de elemento es 1003 por ser un polígono
                                -- exterior.
                        3,      -- La interpretación es 3 por ser un rectángulo optimizado.
                        7,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 7 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1, Z1 -- Valores ordenadas punto 1 (Inicio primer elemento)
                        X2, Y2, Z2 -- Valores ordenadas punto 2 (Fin primer elemento)
                        X3 Y3, Z3 -- Valores ordenadas punto 3 (Inicio segundo elemento)
                        X4, Y4, Z4 -- Valores ordenadas punto 4 (Fin segundo elemento)
                )
        )
)
```

*Figura 35. Ejemplo sentencia de superficie con agujero*

### 3.4.7.2 Composición de superficies tridimensional

En la siguiente figura se puede observar la representación correspondiente a una composición de superficies propia de un sistema tridimensional.

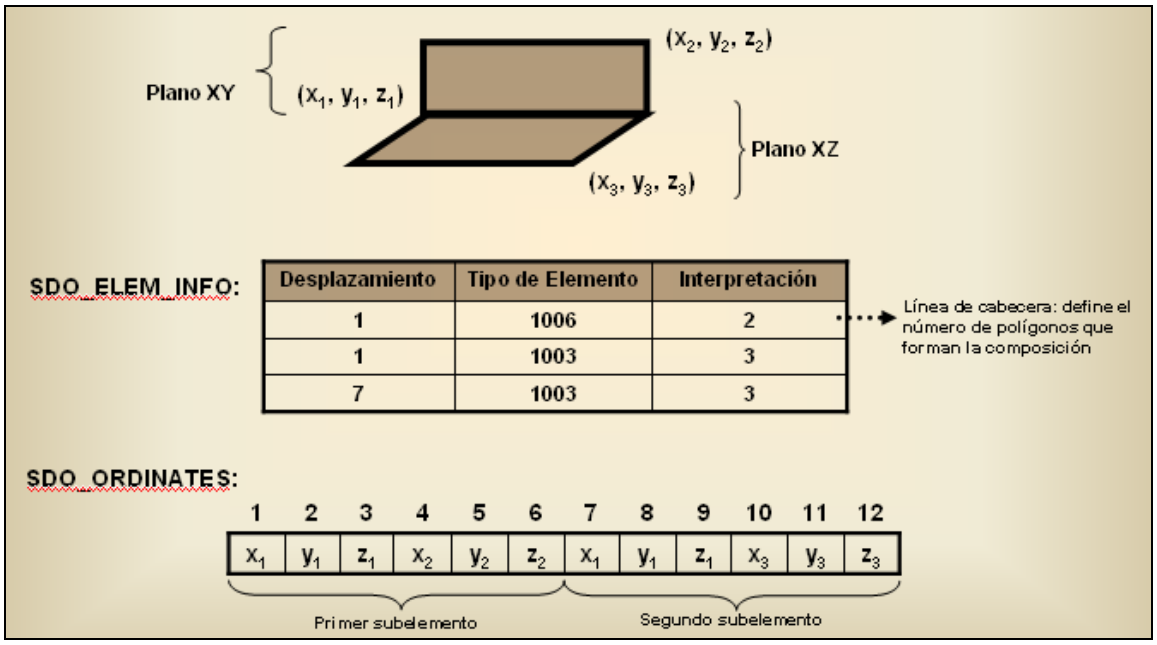



*Figura 36. Ejemplo de composición de superficies (Elaboración propia)*

El orden de los vértices en la composición, no sigue ninguna norma concreta puesto que Oracle no lo estipula, pero si define lo que se conoce como 'superficie normal' usando la regla del pulgar de la mano derecha. Lo veremos con más detalle en el ejemplo de un sólido simple, donde si se establece una norma para la superficie normal.

La sentencia SQL para añadir esta composición con el código '2' y la descripción 'Composición 2' en la tabla 'GEO3D' sería la siguiente:

```
INSERT INTO geo3d VALUES
(       2,    --  código de la geometría, información no espacial
        'Composición 2', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       3003,    -- SDO_GTYPE.  Superficie en tres dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,        -- Línea de cabecera, por definición es 1
                        1006,   -- El tipo de elemento es 1006 por ser una superficie
                                -- compuesta.
                        2,        -- Número de polígonos que forman la composición.
```

```
                        1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                        7,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 7 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                )
                SDO_ORDINATES_ARRAY
                (       X1,Y1, Z1 -- Valores ordenadas punto 1 (Inicio primer elemento)
                        X2, Y2, Z2 -- Valores ordenadas punto 2 (Fin primer elemento)
                        X1 Y1, Z1 -- Valores ordenadas punto 1 (Inicio segundo elemento)
                        X3,Y3, Z3 -- Valores ordenadas punto 3 (Fin segundo elemento)
                )
        )
)
```

*Figura 37. Ejemplo sentencia de composición de superficies*

### 3.4.7.3 Sólido Simple

En la siguiente figura se puede observar la representación correspondiente a sólido simple propio de un sistema tridimensional.

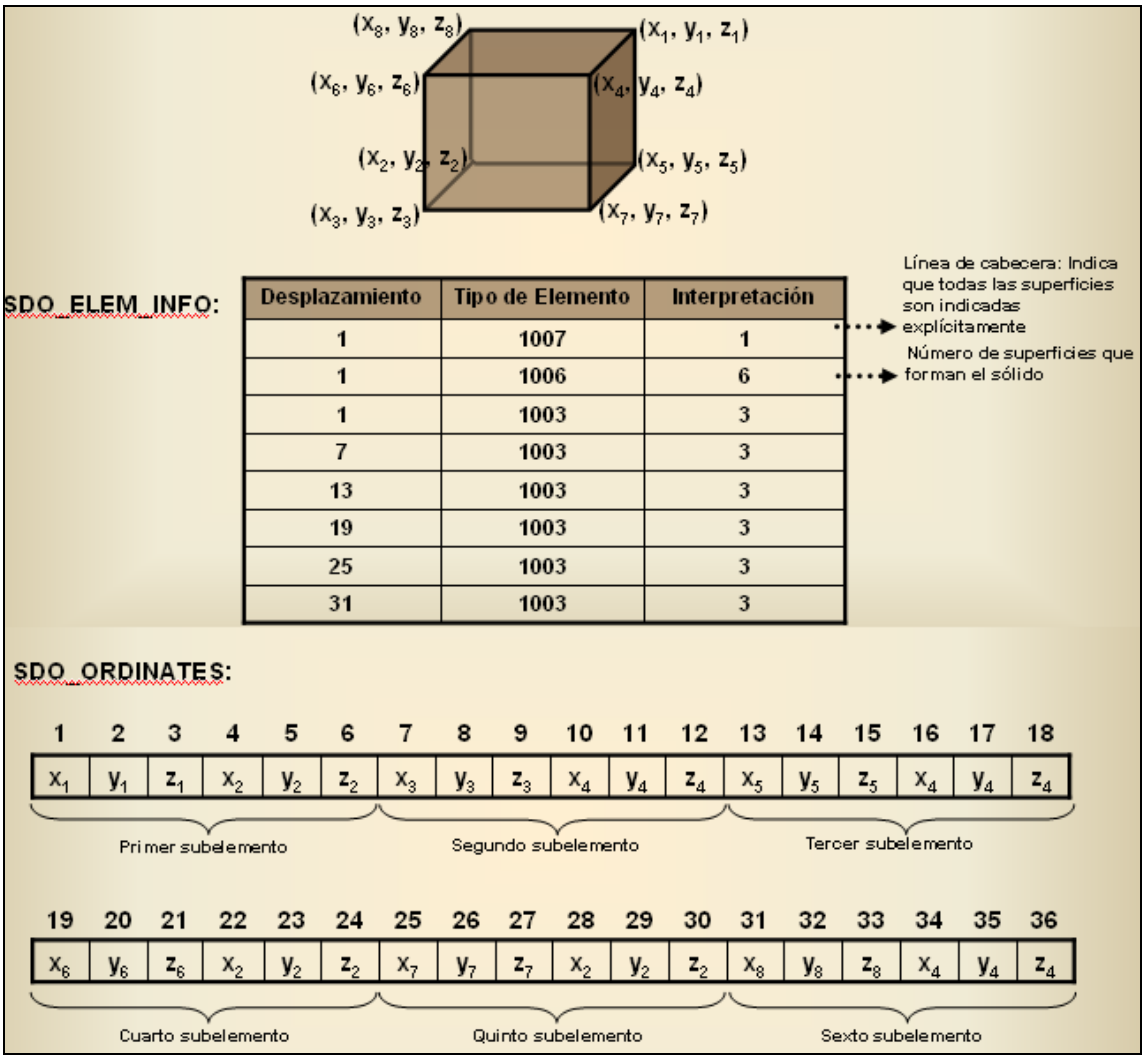



*Figura 38. Ejemplo de sólido simple (Elaboración propia)*

El orden de los vértices en el sólido viene definido por la siguiente regla que establece que la superficie normal para todos los polígonos que forman el sólido tienen que apuntar hacía fuera del sólido.

Curvado los dedos de la mano derecha siguiendo el orden de la los vértices, el dedo pulgar apunta hacia la dirección de la superficie normal.

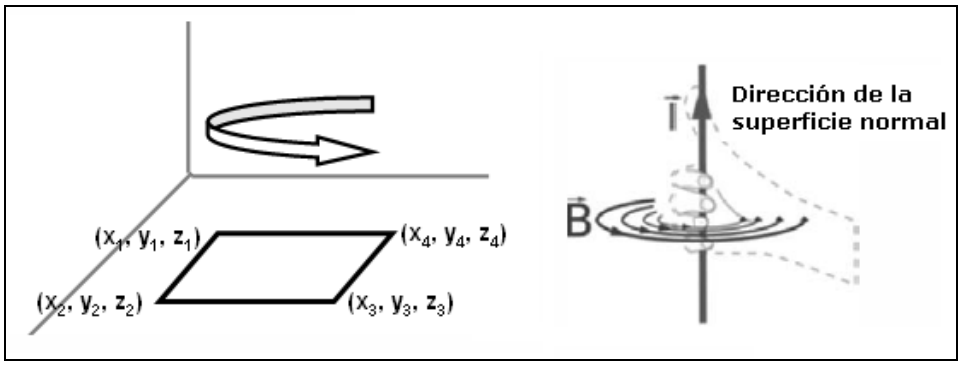



*Figura 39. Dirección de la superficie normal [KGE, 2007]*

En la figura anterior se muestra un rectángulo definido por sus cuatro vértices, pero como hemos visto un rectángulo optimizado puede definirse en base a solo dos vértices, <esquina mínima-esquina máxima> ó <esquina máxima-esquina mínima>.

Usando la siguiente tabla se puede determinar la dirección de la superficie normal para un rectángulo optimizado, en base al orden de sus esquinas y el plano al que es paralelo.

| Paralelo a Plano | Orden esquinas | Dir. Superficie Normal |
|---|---|---|
| XY | Esquina-min, esquina-max | Positivo Eje Z |
| YZ | Esquina-min, esquina-max | Positivo Eje X |
| XZ | Esquina-min, esquina-max | Positivo Eje Y |
| XY | Esquina-max, esquina-min | Negativo Eje Z |
| YZ | Esquina-max, esquina-min | Negativo Eje X |
| XZ | Esquina-max, esquina-min | Negativo Eje Y |

*Tabla 5. Dirección superficie normal para un rectángulo optimizado*

El sólido de la figura "*Figura 38. Ejemplo de sólido simple*" es correcto, dado que la superficie normal de todos los rectángulos optimizados que lo forman apuntan hacía fuera del sólido. A modo de ejemplo, se va a verificar con dos caras opuestas del cubo.

Observando la figura y haciendo uso de la tabla anterior se concluye que:

- Para el rectángulo correspondiente a la base del sólido ($X_7$, $Y_7$, $Z_7$ y $X_2$, $Y_2$, $Z_2$), dado que el orden de las esquinas establecido es <esquina máxima-esquina mínima> y es paralelo al paralelo al XZ, la dirección de la superficie normal es el eje negativo de las Y, que es efectivamente hacía fuera del sólido.

- Para el rectángulo correspondiente a la tapa superior del sólido ($X_8$, $Y_8$, $Z_8$ y $X_4$, $Y_4$, $Z_4$), dado que el orden de las esquinas establecido es <esquina mínima-esquina máxima> y es paralelo al paralelo al XZ, la dirección de la superficie normal es el eje positivo de las Y, que es efectivamente hacía fuera del sólido.

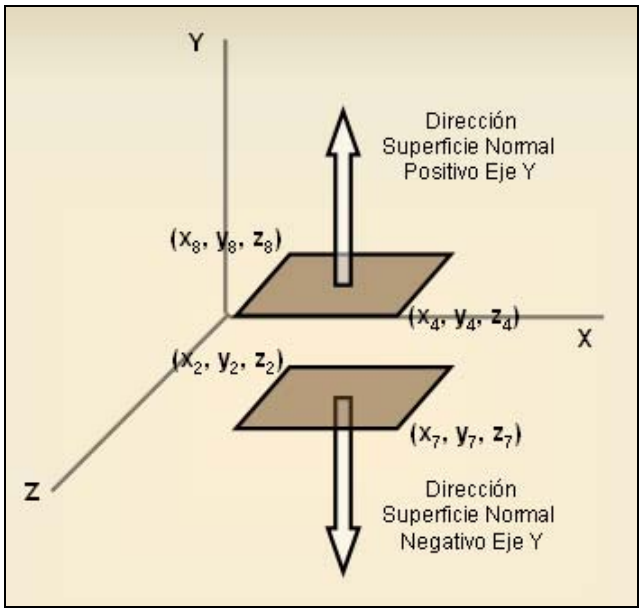



*Figura 40. Dirección superficie normal en un sólido (Elaboración propia)*

Explicado el orden de establecimiento de los vértices en el sólido, se va a ver la sentencia SQL necesaria para insertarlo con el código '3' y la descripción 'Sólido 3' a la tabla 'GEO3D':

```
INSERT INTO geo3d VALUES
(       3,    --  código de la geometría, información no espacial
        'Sólido 3', -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       3008,    -- SDO_GTYPE.  Sólido simple en tres dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
                SDO_ELEM_INFO_ARRAY
                (       1,      -- Línea de cabecera, por definición es 1
                        1007,   -- El tipo de elemento es 1007 por ser un sólido simple
                        1,      -- La interpretación indica que todas las superficies que
                                -- componen el sólido son indicadas explícitamente.
                        1,      -- Línea de cabecera, por definición es 1
                        1006,   -- El tipo de elemento es 1006 por ser una composición
                                de -- superficies
                        6,      -- Número de superficies que forman la composición.
                        1,      -- Las ordenadas del primer elemento que componen la
                                -- geometría comienzan en la posición 1 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                        7,      -- Las ordenadas del segundo elemento que componen la
                                -- geometría comienzan en la posición 7 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
```

```
                        13,     -- Las ordenadas del tercer elemento que componen la
                                -- geometría comienzan en la posición 13 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                        19,     -- Las ordenadas del cuarto elemento que componen la
                                -- geometría comienzan en la posición 19 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                        25,     -- Las ordenadas del quinto elemento que componen la
                                -- geometría comienzan en la posición 25 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                        31,     -- Las ordenadas del sexto elemento que componen la
                                -- geometría comienzan en la posición 31 del array
                        1003,   -- El tipo de elemento es 1003 por ser una Polígono
                                -- exterior.
                        3,      -- La interpretación es 3 porque es un rectángulo
                                -- optimizado.
                )
                SDO_ORDINATES_ARRAY
                (       X1,Y1, Z1 -- Valores ordenadas punto 1 (Inicio primer elemento)
                        X2, Y2, Z2 -- Valores ordenadas punto 2 (Fin primer elemento)
                        X3 Y3, Z3 -- Valores ordenadas punto 3 (Inicio segundo elemento)
                        X4,Y4, Z4 -- Valores ordenadas punto 4 (Fin segundo elemento)
                        X5,Y5, Z5 -- Valores ordenadas punto 5 (Inicio tercer elemento)
                        X4,Y4, Z4 -- Valores ordenadas punto 4 (Fin tercer elemento)
                        X6,Y6, Z6 -- Valores ordenadas punto 6 (Inicio cuarto elemento)
                        X2,Y2, Z2 -- Valores ordenadas punto 2 (Fin cuarto elemento)
                        X7,Y7, Z7 -- Valores ordenadas punto 7 (Inicio quinto elemento)
                        X2,Y2, Z2 -- Valores ordenadas punto 2 (Fin quinto elemento)
                        X8,Y8, Z8 -- Valores ordenadas punto 8 (Inicio sexto elemento)
                        X4,Y4, Z4 -- Valores ordenadas punto 4 (Fin sexto elemento)
                )
        )
)
```

*Figura 41. Ejemplo sentencia de un sólido simple*

## 3.4.8 Ejemplos de colecciones

En los apartados anteriores se han visto ejemplos de geometrías de distintos tipos, las colecciones no son otra cosa que la combinación de estos diferentes tipos de geometrías. Pueden ser homogéneas si son del mismo tipo multipunto, multilínea, multipolígono, multisólido, o heterogéneas en caso contrario.

Siguen las mismas directrices que la construcción que las geometrías compuestas.

### 3.4.8.1 Multilínea bidimensional

En la siguiente figura se puede observar la representación correspondiente a una colección de líneas, en un sistema bidimensional.

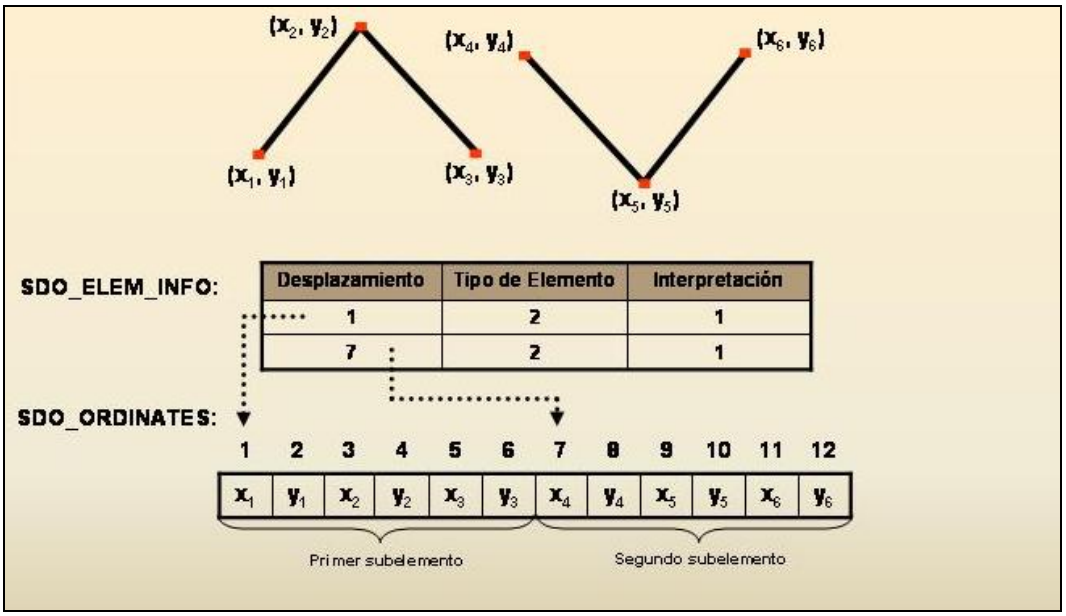



*Figura 42. Ejemplo de multilínea (Elaboración propia)*

La sentencia SQL para añadir esta colección de líneas con el código ‘4’ y la descripción ‘Línea 4’ en la tabla ‘LINEAS’ sería la siguiente:

```
INSERT INTO lineas VALUES
(       4   --  código de la geometría, información no espacial
        ‘Línea 4’, -- descripción  de la geometría, información no espacial
        SDO_GEOMETRY
        (       2006,   -- SDO_GTYPE.  Colección de líneas en dos dimensiones
                NULL, -- SDO_SRID. Sistema cartesiano
                NULL, -- No es un punto por tanto el atributo SDO_POINT debe ser null
```

```
                SDO_ELEM_INFO_ARRAY
                (       1,      -- Las ordenadas del primer elemento que componen la
                                 -- geometría comienzan en la posición 1 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1,      -- La interpretación es 1 por tratarse de vértices
                                 -- conectados por líneas rectas
                        7,      -- Las ordenadas del segundo elemento que componen la
                                 -- geometría comienzan en la posición 7 del array
                        2,      -- El tipo de elemento es 2 por ser una Línea
                        1       -- La interpretación es 1 por tratarse de vértices
                                 -- conectados por líneas rectas
                )
                SDO_ORDINATES_ARRAY
                (       X1, Y1, -- Valores ordenadas punto 1 (Inicio primer elemento)
                        X2, Y2, -- Valores ordenadas punto 2
                        X3 Y3 -- Valores ordenadas punto 3Fin primer elemento)
                        X4, Y4 -- Valores ordenadas punto 4(Inicio segundo elemento)
                        X5, Y5 -- Valores ordenadas punto 5
                        X6, Y6 -- Valores ordenadas punto 6(Fin segundo elemento)
                )
        )
)
```

*Figura 43. Ejemplo sentencia de multilínea*

# 3.5 Metadatos

Como ya se ha mencionado anteriormente, Spatial trata todos los objetos de una columna de tipo SDO_GEOMETRY, como una capa espacial. Para realizar la carga de datos, validación, creación del índice espacial y operaciones de consulta sobre los datos, primero es necesario establecer los metadatos correspondiente a la capa. Estos incluyen la siguiente información:

- Numero de dimensiones.
- Límites superior e inferior de los valares para cada dimensión.
- Tolerancia para cada dimensión.
- Sistema de coordenadas de referencia, para los datos de la capa.

Esta información se almacena en tres vistas, dos de ellas tratadas internamente por Oracle ALL_SDO_GEOM_METADATA, y DBA_SDO_GEOM_METADAT; y la que debe ser tratada por el usuario USER_SDO_GEOM_METADATA. El usuario inserta los datos apropiados en esta última, y Oracle refleja directamente los cambios en las dos primeras.

La estructura física de esta vista, se puede observar en la siguiente figura:

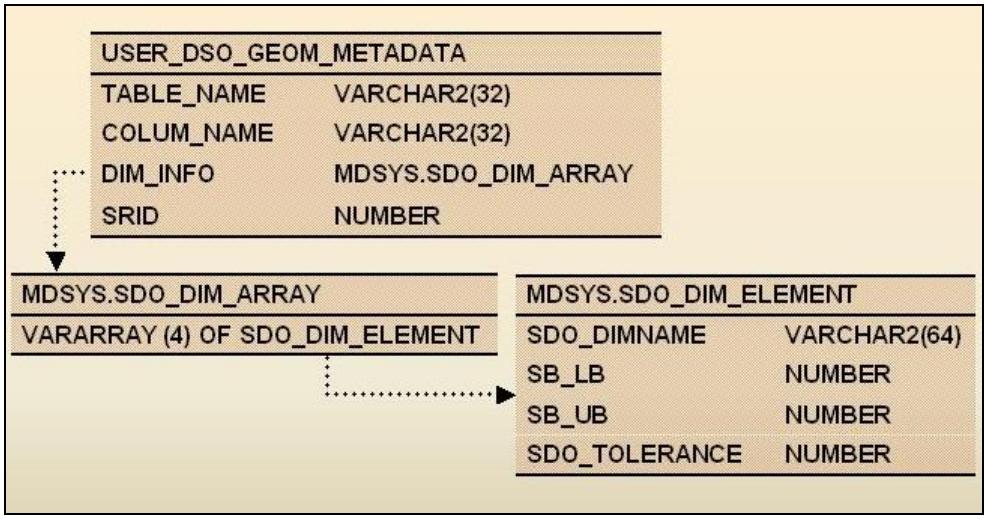



*Figura 44. Vista USER_SDO_GEOM_METADATA (Elaboración propia)*

## 3.5.1 TABLE_NAME, COLUM_NAME

Identifican de manera unívoca cada capa espacial. TABLE_NAME, COLUM_NAME, se corresponden con el nombre de la tabla donde se encuentra la columna de tipo SDO_GEOMETRY, y el nombre de la columna respectivamente. Estos datos siempre son convertidos a mayúsculas antes de ser insertados en la vista.

## 3.5.2 SRID

Este campo al igual que SDO_SRID del tipo SDO_GEOMETRY, especifica el sistema de coordenadas correspondiente a los datos, en este caso en lugar de referirse a un objeto concreto, se refiere a todos los objetos que componen la capa.

En la mayoría de las aplicaciones, no se elige el sistema de coordenadas, sino que se obtiene de los datos de las geometrías que se compran a proveedores de información espacial.

## 3.5.3 DIMINFO

Especifica información sobre cada dimensión de la capa. A continuación se describen los atributos de este tipo de objeto:

### 3.5.3.1 SDO_DIMNAME

Nombre que se le da a la dimensión. El nombre especificado, no es interpretado por Oracle, es decir por ejemplo en un sistema geodésico, a la dimensión que representa la latitud, se le puede llamar ’y’.

### 3.5.3.2 SDO_LB, SDO_UB

Estos dos números, definen los límites inferior y superior para los valores de la dimensión. Estos varían de una aplicación a otra, puesto que dependen de las necesidades específicas de la misma.

### 3.5.3.3 SDO_TOLERANCE

Se usa para expresar el grado de decisión para los datos. Específica la distancia que dos valores deben estar separados para ser considerados diferentes. Si se especifica un valor de '0.5' por ejemplo y la distancia entre dos puntos 'A' y 'B' es menor de esa distancia, se considera que los dos puntos tienen la misma localización.

Por defecto el valor de la tolerancia se mide en la misma medida que los límites inferior y superior. En sistemas geodésicos el valor de la tolerancia se mide siempre en metros. Oracle requiere que el valor de la tolerancia sea la misma para todas las dimensiones.

Los índices espaciales y otras operaciones a nivel de capa, usan la información almacenada en DIMINFO y el valor de la tolerancia expresada en este atributo.

Un segundo uso de la tolerancia es el de las funciones espaciales. La mayoría de estas funciones no leen el valor almacenado en los metadatos de la capa, sino que lo espera como un parámetro de entrada en la función. Si no se fija un valor adecuado, este puede causar resultados inesperados o incorrectos.

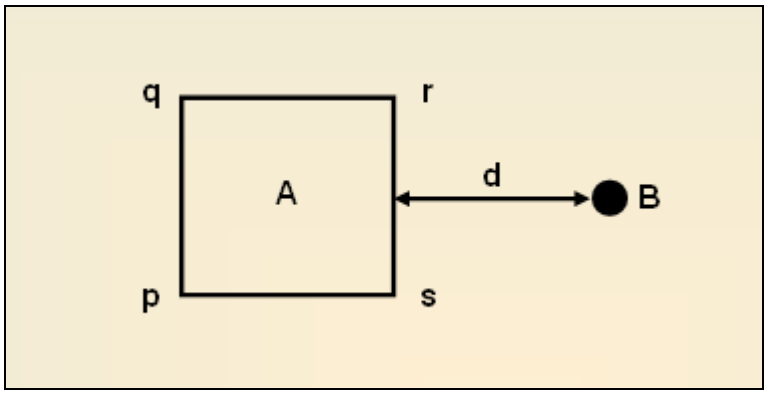



*Figura 45. Significado de la Tolerancia (Elaboración propia)*

Considérense los objetos 'A' y 'B' separados una distancia 'd'. La relación espacial entre ellos y que el objeto A sea considerado una geometría válida, depende del valor que se le dé a la tolerancia:

- Relación entre las geometrías: si la distancia es menor que el valor de la tolerancia, se considera que 'B' es un punto del borde exterior de 'A', es decir, B tiene intersección con A. Por el contrario si d es mayor o igual que el valor de la tolerancia 'A' y 'B' son disjuntos, es decir no tienen intersección.

- Validación de A: si la distancia entre un vértice de rectángulo y cualquier otro vértice del mismo es menor que la tolerancia, se consideran vértices duplicados y por tanto la geometría no es válida. Por el contrario si la

distancia entre los vértices es mayor que la tolerancia, se consideran vértices distintos y por tanto la geometría es correcta.

Como se puede observar en el ejemplo anterior, el valor de la tolerancia juega un papel muy importante.

Como regla general, el valor de la tolerancia debería establecerse como la distancia más pequeña perceptible en nuestra aplicación. En la mayoría de las aplicaciones esta distancia corresponde a la mitad de la diferencia entre dos coordenadas individuales. Esta técnica puede ser aplicada en sistemas locales, como CAD/CAM, o sistemas proyectados. Sin embargo para localizaciones en la superficie terrestre usando sistemas de coordenadas geodésicos, la diferencia en los valores de longitud y latitud de dos localizaciones no se corresponde con la actual distancia entre ellos. En estos casos las ordenadas son interpretadas en grados y la tolerancia en metros.

| Sistema de coordenadas | Valores SRID | Tolerancia | Unidades |
|---|---|---|---|
| Geodésico | Select SRID from MDSYS.CS_SRS where WKTEXT like '%GEOGCS%' | 0.5 (no debería ser menor de 0.05) | Metros para la tolerancia, grados para las dimensiones. |
| Proyectado | Select SRID from MDSYS.CS_SRS where WKTEXT like ' %PROJCS%' | La mitad de la diferencia menor entre dos valores cualesquiera de la dimensión. | Las unidades para la tolerancia son las mismas que para las dimensiones |
| Local | Select SRID from MDSYS.CS_SRS where WKTEXT like '%LOCAL_CS%' | La mitad de la diferencia menor entre dos valores cualesquiera de la dimensión. | Las unidades para la tolerancia son las mismas que para las dimensiones |
| No especificado | NULL | La mitad de la diferencia menor entre dos valores cualesquiera de la dimensión. | Las unidades para la tolerancia son las mismas que para las dimensiones |

*Tabla 6. Valores de tolerancia recomendados*

Como ejemplo, se van a insertar los metadatos para las tablas 'PUNTOS', 'LINEAS' Y 'POLIGONOS'.

```
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES ('PUNTOS', 'GEOMETRIA',
    MDSYS.SDO_DIM_ARRAY
      (MDSYS.SDO_DIM_ELEMENT('X', 0, 100, 0.25),
       MDSYS.SDO_DIM_ELEMENT('Y', 0, 100, 0.25)
      ),
      262144);
```

```
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES ('LINEAS', 'GEOMETRIA',
    MDSYS.SDO_DIM_ARRAY
      (MDSYS.SDO_DIM_ELEMENT('X', 0, 100, 0.25),
       MDSYS.SDO_DIM_ELEMENT('Y', 0, 100, 0.25)
      ),
      262144);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
DIMINFO, SRID)
  VALUES ('POLIGONOS', 'GEOMETRIA',
    MDSYS.SDO_DIM_ARRAY
      (MDSYS.SDO_DIM_ELEMENT('X', 0, 100, 0.25),
       MDSYS.SDO_DIM_ELEMENT('Y', 0, 100, 0.25)
      ),
      262144);
```

*Figura 46. Metadatos capas puntos, líneas y polígonos*

Estos metadatos servirán de base en los ejemplos de validación, creación del índice espacial y operaciones de consulta sobre los datos, que se verán más adelante en los siguientes apartados.

Se trata de un escenario en dos dimensiones ‘X’ e ‘Y’, cuyas coordenadas deben estar dentro de los límites de ‘0’ a ‘100’ para ambas dimensiones y cuyo sistema de referencia es el sistema local con SRID ‘262144’. Para los escenarios de trabajo que se van a utilizar en los siguientes apartados se ha creído conveniente establecer el valor de la tolerancia a ‘0.25’.

# 3.6 Carga de datos espaciales

Una vez definidos el tipo de objeto utilizado por Oracle para almacenar la información espacial, la creación de capas consistentes en tablas con columnas de tipo SDO_GEOMETRY y la introducción de metadatos asociados a la capa, se van a ver los mecanismos que proporciona Oracle para cargar los datos en las capas.

En los ejemplos de geometrías simples, compuestas y en tres dimensiones de los apartados “*3.4.5 Ejemplos de geometrías simples*”, “*3.4.6 Ejemplos de geometrías compuestas*”, “*3.4.7 Ejemplos de geometrías en tres dimensiones*” se ha podido observar la introducción de datos en columnas de tipo SDO_GEOMETRY mediante sentencias ‘INSERT’. Este método, no es el utilizado en aplicaciones reales, puesto que puede consumir mucho tiempo y aumenta la posibilidad de cometer errores en la carga de la información.

A continuación se van a resumir los distintos mecanismos de carga de datos que nos propone Oracle.

## 3.6.1 Carga desde ficheros de texto usando SQL Loader

SQL Loader es una utilidad de Oracle que permite cargar datos de forma masiva desde un fichero de control, no es propia de Spatial También es usada con información no espacial.

Un fichero de control, es un fichero de texto con extensión 'ctl', que se divide en dos partes diferenciadas. La parte que describe la tabla y la estructura de los datos que se van a insertar en la misma; y las líneas de registros correspondientes a los datos de carga.

El siguiente fichero de control, contiene la estructura e información necesarias, para cargar 4 registros en la columna "geometria" de la tabla 'LINEAS'.

```
LOAD DATA
INFILE *
CONTINUEIF NEXT(1:1)='#'
INTO TABLE lineas
FIELDS TERMINATED BY '|'
TRAILING NULLCOLS (
cd_linea INTEGER EXTERNAL,
descripcion CHAR,
geometria COLUMN OBJECT
(
        SDO_GTYPE INTEGER EXTERNAL,
        SDO_SRID INTEGER EXTERNAL,
        SDO_ELEM_INFO VARRAY terminated by '/' (elementos FLOAT EXTERNAL),
        SDO_ORDINATES VARRAY terminated by '/' (ordenadas FLOAT EXTERNAL)

)
}
BEGINDATA
 1|Linea 1|2002|262144|
#1|2|1|/
#5|3|6|4|6|7|8,5|7|/
 2|Linea 2|2002|262144|
#1|2|1|/
#10,5|1,5|12|1,5|/
 3|Linea 3|2002|262144|
#1|2|1|/
#10,5|1,5|12|1,5|/
 4|Linea 4|2002|262144|
#1|2|1|/
#3|6|7|1|7|1|/
```

*Figura 47. Ejemplo de carga con SQL Loader*

Puesto que el manejo de ficheros de control no es algo propio de Spatial, solo se va a hacer una breve mención del significado de las distintas líneas del archivo:

- LOAD DATA: Línea requerida siempre en el fichero de control
- INIFILE *: Indica si los datos aparecen en el mismo archivo (*) o en un archivo a parte (fichero con extensión 'dat').En el caso de separar los datos en otro fichero, la línea 'BEGIN DATA' y las correspondientes a la información

de los registros, se eliminan del fichero de control, y se incluyen solo las líneas de datos en el fichero ‘dat’.

- INFO TABLE líneas: Indica el nombre de la tabla que va a recibir los datos. En este caso la tabla ‘LINEAS’, que suponemos que está vacía puesto que si tuviera datos se debería usar ‘APPEND INFO TABLE’ en lugar de ‘INFO TABLE’.

- FIELDS TERMINATED BY '|’: Indica el separador de los valores de las diferentes columnas de cada registro. En este caso '|'.

- TRAILING NULLCOLS: Las columnas que no estén definidas entre los paréntesis que siguen a esta expresión, serán actualizados automáticamente con el valor NULL.

- BEGIN DATA: Indica que en las líneas siguientes aparecen los registros a insertar en la tabla.

Un detalle importante a tener en cuenta, es que SQL Loader no permite procesar registros de más de 64KB, si los datos están incluidos en el fichero de control; y de 1MB si están separados en el fichero de datos. Obsérvese en el fichero de control la línea ‘CONTINUEIF NEXT(1:1) = ‘#’’. Esta permite tratar los registros separados en distintas líneas, de modo que, cada vez que se procesa una línea, si el primer carácter de la misma es ‘#’, significa que es continuación de un registro y no uno nuevo. En el caso que nos ocupa, no sería necesario, solo se ha incluido para mostrar su uso.

Tras haber definido el fichero de control, el siguiente paso será la ejecución del SQL Loader para realizar el proceso de carga. La línea de comando siguiente carga los datos del fichero ‘LINEAS.ctl’ en la instancia ‘BDPFC’.

```
sqlldr system/manager@BDPFC control=LINEAS.ctl log=LINEAS.log
```

*Figura 48. Ejecución carga SQL Loader*

Se añade el parámetro ‘log’ para que se cree en un fichero ‘LINEAS.log’ con el resumen de la estructura de los registros incluidos en el fichero y el resultado de la operación. De modo que los posibles errores derivados de un formato incorrecto del fichero queden registrados en el mismo. El siguiente texto muestra el contenido del fichero log resultante de ejecutar la anterior línea de comando.

```
SQL*Loader: Release 9.2.0.1.0 - Production on Lun Abr 25 20:41:36 2011

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

  Archivo de Control:   LINEAS.ctl
  Archivo de Datos:      LINEAS.ctl
  Archivo Defectuoso:     LINEAS.bad
  Desechar Archivo:  ninguno especificado

 (Permitir todos los registros desechados)
```

```
Número a cargar: ALL
Número a ignorar: 0
Errores permitidos: 50
Matriz de enlace:     64 filas, máximo de 256000 bytes
Continuación:   1:1 = 0X23(carácter '#'), en el siguiente registro físico
Ruta de acceso utilizada:      Convencional

Tabla LINEAS, cargada de cada registro lógico.
Opción INSERT activa para esta tabla: INSERT
Opción TRAILING NULLCOLS activada

 Nombre Columna             Posición   Long  Term Entorno Tipo de dato
------------------------------ ---------- ----- ---- ---- ---------------------
CD_LINEA                       FIRST     *   |      CHARACTER
DESCRIPCION                      NEXT     *   |      CHARACTER
GEOMETRIA                      DERIVED     *          COLUMN OBJECT

*** Campos en GEOMETRIA
SDO_GTYPE                        NEXT     *   |      CHARACTER
SDO_SRID                       NEXT     *   |      CHARACTER
SDO_ELEM_INFO                    DERIVED     *   /      VARRAY

*** Campos en GEOMETRIA.SDO_ELEM_INFO
ELEMENTOS                      FIRST     *   |      CHARACTER
*** Fin de campos en GEOMETRIA.SDO_ELEM_INFO

SDO_ORDINATES                    DERIVED     *   /      VARRAY

*** Campos en GEOMETRIA.SDO_ORDINATES
ORDENADAS                      FIRST     *   |      CHARACTER
*** Fin de campos en GEOMETRIA.SDO_ORDINATES

*** Fin de campos en GEOMETRIA

Tabla LINEAS:
  4 Filas se ha cargado correctamente.
  0 Filas no cargada debido a errores de datos.
  0 Filas no cargada porque todas las cláusulas WHEN han fallado.
  0 Filas no cargada porque todos los campos eran nulos.

Espacio asignado a matriz de enlace:          116096 bytes (64 filas)
Bytes de buffer de lectura: 1048576

Total de registros lógicos ignorados:         0
Total de registros lógicos leídos:          4
Total de registros lógicos rechazados:        0
Total de registros lógicos desechados:        0

La ejecución empezó en Lun Abr 25 20:41:36 2011
La ejecución terminó en Lun Abr 25 20:41:36 2011

Tiempo transcurrido:     00:00:00.14
Tiempo de CPU:         00:00:00.08
```

*Figura 49. Fichero log de carga del SQL Loader*

Aunque todavía no se ha explicado cómo se añaden índices espaciales a las tablas, es importante destacar, que antes de realizar la carga de datos, es conveniente, borrar los

índices, ya que sino el proceso puede ser muy lento. Una vez finalizada la carga, se volverían a incluir.

## 3.6.2 Migración de datos

Oracle proporciona varias formas de migrar datos espaciales entre diferentes bases de datos. Estas incluyen las utilidades utilizadas en Oracle para elaborar copias de seguridad con posibilidad de importar y exportar datos como son Import/Export y la unidad de migración a partir del tablespace.

La forma más sencilla de migrar o importar/exportar datos, es a través de los ficheros '.dmp'. Estos se manejan a través de las utilidades Import /Export y tienen el mismo funcionamiento que para datos no espaciales.

Por ejemplo, se pueden exportar los datos de la tabla 'LINEAS', cuya carga se mostró en el punto anterior, del esquema SPATIAL en el que fue creado al fichero 'LINEAS.dmp' mediante la siguiente línea de comando:

```
EXP spatial/spatial FILE=LINEAS.bmp TABLES=LINEAS
```

*Figura 50. Ejemplo exportación tabla*

Tras obtener el fichero con los datos de la exportación, se puede importar a otro esquema distinto usando la siguiente línea de comando:

```
IMP scott/tiger FILE =LINEAS.dmp IGNORE=y INDEXES=N
```

*Figura 51. Ejemplo importación tabla*

En la línea de comando, se pueden establecer una serie de parámetros, para personalizar ciertos aspectos de la importación. El parámetro 'IGNORE=Y' indica que se ignoren los avisos producidos porque algunos de los elementos que se estén importando ya existan en el esquema de destino. Con 'INDEXES=N' únicamente se importan los datos de las tablas, no los índices existentes en las mismas. Si no se especificase este último parámetro, si existiera un índice en la columna 'geometria' de la tabla 'LINEAS' antes de ser exportada, sería creado automáticamente después de la importación de los datos. A su vez la creación automática del índice provocará también de forma directa la inserción de los metadatos correspondientes a la capa en la vista USER_SDO_GEOM_METADATA. Para ello utiliza los metadatos de la base de datos exportada.

En los ejemplos anteriores, sólo se han mencionado las utilidades estándar. Pero existen otros componentes de Oracle con utilidades de migración de datos más eficientes. En el caso de elegir este mecanismo para realizar una migración de datos real en la que la el volumen de información a migrar sea considerable, habría que tener en cuenta la existencia de los mismos.

Hasta el momento, se ha planteado la migración de un conjunto de tablas. ¿Qué ocurre si se quiere transportar un tablespace entero? Por ejemplo, en el supuesto que se

quiera migrar el conjunto de tablas que componen el tablespace ‘TBS’. Los pasos a seguir serían los siguientes:

1.- Generar el fichero de exportación desde la base de datos de origen:

```
SQLPLUS spatial/spatial
EXECUTE SDO_UTIL.PREPARE_FOR_TTS('TBS');
CONNECT SYSTEM/MANAGER AS SYSDBA
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK('TBS', TRUE);
ALTER TABLESPACE TBS READ ONLY;
EXIT;
EXP USERID = '"SYS/<password>"' TRANSPORT_TABLESPACE=Y
TABLESPACES=TBS
FILE=trans_ts.dmp
```

*Figura 52. Ejemplo de generación de fichero de exportación de un tablespace*

Este script crea el fichero ‘trans_ts.dmp’ con los metadatos del tablespace. Se deben copiar éste y el fichero ‘sdo_tts.dbf’ al sistema de base de datos de destino.

2.- Importar el fichero de exportación en la base de datos destino: tras copiar los datos del tablespace al sistema de base de datos de destino, habrá que ejecutar la sentencia de importación:

```
IMP USERID = '"SYS/<password>"' TRANSPORT_TABLESPACE=Y
FILE=trans_ts.dmp
DATAFILES='sdo_tts.dbf' TABLESPACES=tbs
```

*Figura 53. Ejemplo de importación de un tablespace*

Este script crea el tablespace e introduce su contenido en la base de datos de destino.

2.- Habilitar los índices espaciales: Por último una vez importado en tablespace, es necesario modificar el tablespace para permitir operaciones tanto de lectura como de escritura, y ejecutar el procedimiento SDO_UTIL.INITIALIZE_INDEXES_FOR_TTS para habilitar los índices espaciales.

```
SQLPLUS SYS/<password>
ALTER TABLESPACE TBS READ WRITE;
CONNECT spatial/spatial;
EXEC SDO_UTIL.INITIALIZE_INDEXES_FOR_TTS;
```

*Figura 54. Ejemplo de habilitación de  índices espaciales*

El procedimiento INITIALIZE_INDEXES_FOR_TTS activa los índices espaciales existentes en las tablas del tablespace importado. Sin embargo, sólo funcionará si el formato endian de los sistemas operativos de la base de datos origen y destino son las mismas. Es decir, los sistemas operativos de las bases de datos de origen y destino deben usar la misma forma para almacenar números de más de un byte. Si los datos numéricos de origen están expresados en Big Endian (MSB LSB), el sistema operativo de las base de datos de destino debe usar también el formato Big Endian. Lo mismo se aplica al formato Little Endian (LSB MSB).
Si esto no se cumple, los índices deben reconstruirse uno por uno con el comando ALTER INDEX REBUILD.

## 3.6.3 Migración desde versiones anteriores de Oracle Spatial

El tipo SDO_GEOMETRY ha evolucionado desde sus primeras versiones y lo seguirá haciendo en un futuro. Por ello, Spatial incluye el paquete SDO_MIGRATE, con funciones como TO_CURRENT, que permite convertir datos espaciales de versiones anteriores a la versión actual desde la que se está usando la función.

Por ejemplo ejecutando la sentencia:

```
EXECUTE SDO_MIGRATE.TO_CURRENT('lineas', 'geometria', 100);
```

*Figura 55. Ejemplo migración desde una versión anterior*

Se migrarían los datos de la columna 'GEOMETRIA' de la tabla 'LINEAS' a la versión actual, consolidando los cambios cada 100 registros.

El paquete SDO_MIGRATE presenta otras posibilidades, como la conversión de un solo registro de una columna en lugar de todos los registros de la tabla. En el documento 'Oracle Spatial User´s Guide and Reference', se pueden consultar todas las posibilidades que ofrece el paquete.

## 3.6.4 Carga de datos desde fuentes externas

Las aplicaciones de negocio que operan sobre datos espaciales comunes que representan características geográficas, como la localización de una ciudad, el trazado de una carretera, normalmente obtienen la información de distribuidores SIG y agencias de mapeo geográfico como. NAVTEQ. Los formatos utilizados por estos, no son interpretados por Spatial, pero existen una variedad de conversores disponibles en el mercado que realizan la conversión de unos formatos a otros.

### 3.6.4.1 Utilidad SHP2SDO

Para entender el funcionamiento de estos conversores, se va a mostrar un ejemplo de la utilidad SHP2SDO de Oracle, disponible de forma gratuita, pero para la que no dan soporte. Este conversor toma como entrada ficheros ESRI, utilizados por sistemas de información geográfica (SIG) como Arc/Info o ArcGIS, y proporciona ficheros de salida para cargar mediante el SQL Loader.

Este ejemplo se basa en datos geodésicos por lo que no sirve la tabla 'LINEAS' utilizada hasta ahora en los ejemplos anteriores. Por tanto, supóngase el caso de una aplicación de negocio que debe trabajar con datos referentes a todas las ciudades de España, y se han adquirido los datos en formato ESRI. La siguiente línea de comando ilustra un ejemplo de conversión de los mismos:

```
SHP2SDO ciudades –g localizacion –x(-180, 180) –y (-90, 90)  -s 8307 -t 0.5
```

*Figura 56. Ejemplo de conversión de datos mediante la utilidad SHP2SDO*

Como se puede observar, la utilidad va a recibir como parámetros:

- Los datos en formato ESRI a convertir. En dicho formato la información se encuentra repartida en tres ficheros diferentes ‘ciudades.shp’, ‘ciudades.shx’ y ‘ciudades.dbf’.
- El nombre de la columna que contiene el dato espacial.
- Los valores máximos y mínimos posibles para las dimensiones ‘x’ y ‘y’.
- El sistema de coordenadas de los datos.
- Y el valor de la tolerancia.

Ejecutando esta línea de comandos, se generarán tres ficheros de salida:

- El fichero ‘ciudades.sql’ con las sentencias SQL para generar la tabla y los metadatos de la misma
- El fichero de control de SQL Loader ‘ciudades.ctl’.
- Y el fichero de datos ‘ciudades.dat’ a cargar.

A partir de estos ficheros, se pueden cargar los datos con la utilidad SQL Loader vista anteriormente en este mismo apartado.

### 3.6.4.2 Formato WKT

Otro formato muy importante a tener en cuenta, es el correspondiente al estándar internacional SQL/MM. Este estándar, define, entre otras cosas, los tipos de datos necesarios para representar características espaciales.

El tipo de datos SDO_GEOMETRY de Spatial, modela la mayoría de estos tipos y presenta constructores que permiten convertir las instancias de los objetos de tipo SDO_GEOMETRY, en los formatos definidos por el estándar, WKT (texto) y WKB (binario).

La conversión al formato WKT se realiza mediante el constructor GET_WKT(), que devuelve un tipo CLOB. De forma análoga, la conversión al tipo WKB, se realiza mediante el constructor GET_WKB(), y en este caso devuelve un tipo BLOB. La siguiente sentencia muestra un ejemplo de uso:

```
SELECT A.GEOMETRIA.GET_WKT() WKT
FROM LINEAS
WHERE CD_LINEA = 1;

WKT
-----------------------------------------------
LINESTRING (5.0 3.0, 6.0 4.0, 6.0 7.0, 8.5 7.0)
```

*Figura 57. Ejemplo de conversión al formato WKT*

Puesto que WKT y WKB son estándares soportados por muchos distribuidores de datos espaciales, los constructores mencionados anteriormente, posibilitan la transformación de datos en formato Spatial a los formatos externos usados por éstos.

### 3.6.4.3 Formato GML

Por último, otro formato muy utilizado es el correspondiente al lenguaje GML (Geographic Markup Language), basado en XML. Existen varias versiones de GML, GML 2.0 soporta sólo datos en dos dimensiones, mientras que GML3.1.1 da soporte también a datos en tres dimensiones. Las funciones de conversión a utilizar para realizar la conversión de objetos SDO_GEOMETRY a formato GML, depende de la versión que se utilice. Por ejemplo si se trabaja con la versión GML 2.0 debe usarse la función TO_GMLGEOMETRY:

```
SELECT TO_CHAR(SDO_UTIL.TO_GMLGEOMETRY(GEOMETRIA)) formato_gml
FROM LINEAS
WHERE CD_LINEA = 1;

FORMATO_GML
--------------------------------------------------------------------------------
<gml:LineString srsName="SDO: 262144" mlns:gml="http://www.opengis.net/gml">
    <gml:coordinates decimal="." cs="," ts=" ">5,3,6,4,6,7,8.5,7 </gml:coordinates>
</gml:LineString>
```

*Figura 58. Ejemplo de conversión al formato GML*

La función TO_GMLGEOMETRY recibe como entrada una instancia del tipo SDO_GEOMETRY, y devuelve un tipo CLOB. Este objeto contiene información sobre el tipo de geometría, el sistema de coordenadas y las coordenadas de la misma, expresada con las etiquetas propias de GML.

A partir de la versión de Oracle 11g también se puede realizar el proceso inverso, es decir partiendo de una geometría en formato GML obtener el objeto SDO_GEOMETRY correspondiente, la función a utilizar también dependerá de la versión de GML. Para GML 2.0 usaremos la función FROM_GMLGEOMETRY:

```
SDO_UTIL.FROM_GMLGEOMETRY(
TO_CLOB(
'<gml:LineString srsName="SDO: 262144" mlns:gml="http://www.opengis.net/gml">
    <gml:coordinates decimal="." cs="," ts=" ">5,3 6,4 6,7 8.5,7 </gml:coordinates>
</gml:LineString>'
  )
) GEOM FROM DUAL;
--------------------------------------------------------------------------------
SDO_GEOMETRY(2002, 262144, NULL,
SDO_ELEM_INFO_ARRAY(1, 2, 1),
SDO_ORDINATE_ARRAY(5,3,6,4,6,7,8.5,7)
)
```

*Figura 59. Ejemplo de conversión de GML a SDO_GEOMETRY*

La función FROM_GMLGEOMETRY recibe como entrada un CLOB con la información de la geometría en formato GML y devuelve el tipo SDO_GEOMETRY correspondiente.

En el documento 'Oracle XML Database Developer´s Guide' se pueden encontrar todas las funciones disponibles para realizar la conversión según el número de geometrías a tratar y el tipo de las mismas.

# 3.7 Validación y depuración

En el apartado anterior se han explicado diferentes formas de cargar los datos espaciales en las columnas SDO_GEOMETRY de las tablas. Una vez realizado el proceso de carga de datos, es necesario validar si corresponden a geometrías correctas en Oracle Spatial. Realizar la validación y depuración de los datos es un proceso muy importante, dado que el provecho que se obtenga del análisis de los datos dependerá lógicamente de la calidad de los mismos.

A continuación se presentan los mecanismos de validación y depuración de datos espaciales disponibles en Oracle.

## 3.7.1 Validación

Oracle proporciona la función VALIDATE_GEOMETRY_WITH_CONTEXT aplicado a una única geometría y el procedimiento VALIDATE_LAYER_WITH_CONTEXT utilizado cuando se quiera validar una capa entera. Ambos operan sobre geometrías en dos o tres dimensiones y devuelven una cadena de caracteres con la descripción del error, cuando encuentra geometrías no válidas. Forman parte del paquete SDO_GEOM de Oracle.

### 3.7.1.1 VALIDATE_GEOMETRY_WITH_CONTEXT

Es la función a utilizar para validar una única geometría y presenta las dos variantes que se muestran a continuación:

```
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
(
        geometry IN SDO_GEOMETRY,
        tolerance IN NUMBER
) RETURN VARCHAR2;
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
(
        geometry IN SDO_GEOMETRY,
        diminfo IN SDO_DIM_ARRAY
) RETURN VARCHAR2;
```

*Figura 60. Sintaxis función VALIDATE_GEOMETRY_WITH_CONTEXT*

Ambas funciones reciben el objeto SDO_GEOMETRY a validar y un segundo parámetro consistente en el valor de la tolerancia utilizado en la validación ó la información de los límites y el valor de la tolerancia para cada una de las dimensiones (diminfo). La ejecución de las funciones devuelve una cadena de caracteres con el valor 'TRUE' si la geometría es correcta, el código de error de Oracle si la geometría no es válida y el error es conocido o 'FALSE' en caso de que la geometría no sea válida y el error no pueda identificarse.

### 3.7.1.2 VALIDATE_LAYER_WITH_CONTEXT

Se utilizará este procedimiento cuando sea necesario validar los datos de una capa entera. Su sintaxis es la siguiente:

```
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT
(
        table_name IN VARCHAR2,
        column_name IN VARCHAR2,
        result_table IN VARCHAR2
        [,commit_interval IN NUMBER]
)
```

*Figura 61. Sintaxis función VALIDATE_LAYER_WITH_CONTEXT*

Los parámetros 'table_name' y 'column_name' identifican la capa espacial a validar. El resultado de ejecutar el proceso de validación, se recoge en la tabla cuyo nombre se indique en el parámetro 'result_table'. Dicha tabla debe haberse creado antes de ejecutar la llamada al procedimiento y debe tener la siguiente estructura:

```
SDO_ROWID ROWID
STATUS VARCHAR2(2000)
```

*Figura 62. Estructura tabla resultados de la validación*

En el campo SDO_ROWID se inserta el ROWID de la geometría y en el campo STATUS el resultado de la validación. El último parámetro indica cada cuantas geometrías validadas se realiza el commit en la tabla 'result_table'.

### 3.7.1.3 Criterios de validación

Vistas las funciones de validación proporcionadas por Oracle, cabe preguntarse qué criterios de validación aplican para decidir si una geometría es válida.

En primer lugar Oracle analiza el atributo SDO_GTYPE que es el que establece el tipo de geometría que representa el objeto. Posteriormente dependiendo del tipo de geometría también analizará el atributo SDO_ETYPE de cada uno de los elementos que componen la geometría.

Por último se aplican diferentes reglas de validación, según las características topográficas del tipo de geometría identificado a través de análisis de los atributos SDO_GTYPE y SDO_ETYPE.

Los siguientes apartados contienen un resumen de restricciones a cumplir por tipo de geometría.

#### 3.7.1.3.1 Puntos

Un punto es válido cuando sus coordenadas están dentro de los límites establecidos en el atributo DIMINFO, que recordemos que contiene la información de las dimensiones en las que está representada la geometría.

#### 3.7.1.3.2 Líneas

Una línea debe satisfacer las siguientes reglas de validación:

- Todos los puntos que la forman son distintos.
- Está formada al menos por dos puntos.

#### 3.7.1.3.3 Polígonos

Cada anillo que forma el polígono debe cumplir las siguientes restricciones:

- Cerrado: El primer y último vértice del anillo coinciden, es decir son idénticos.
- Planaridad: Todos los vértices del anillo están en el mismo plano (dentro del valor de la tolerancia)
- Bordes sin cruces: Los bordes de un anillo no se pueden cruzar entre sí.
- Línea válida: el anillo es una línea válida, según las restricciones establecidas para las líneas.
- Co-planaridad de los anillos: dado que le polígono define un área en un plano, todo los anillos deben estar en el mismo plano (dentro del valor de la tolerancia).
- Orientación apropiada: El orden de los vértices de los anillos internos debe ser opuesto al de los anillos externos.
- Área contigua: El anillo exterior juntos con los interiores, si los hay, definen una única área contigua. Esto implica que los anillos internos no pueden dividir al polígono en áreas disjuntas.
- Anillos no superpuestos: Lo anillos son pueden superponerse (dentro del valor de la tolerancia), pueden tocarse en un punto, siempre y cuando no violen la restricción de dividir al polígono en áreas disjuntas.
- Conexión anillos internos: Cada anillo interno debe estar dentro del anillo externo (dentro del valor de la tolerancia), puede tocarlo en un punto siempre y cuando no viole la restricción de dividir al polígono en áreas disjuntas.

- Para polígonos en dos dimensiones, el orden de los vértices del anillo exterior se establece en contra del avance de las agujas de un reloj, mientras que el de los anillos internos se establece en el mismo sentido del avance de las agujas de un reloj.

- Para rectángulos optimizados en tres dimensiones, si el anillo exterior de define por <esquina mínima - esquina máxima>, el interior debe hacerse al contrario, es decir, <esquina máxima – esquina mínima> (o viceversa).

#### 3.7.1.3.4 Composición de superficies

Una composición de superficies debe satisfacer las siguientes reglas de validación:

- Polígonos válidos: Cada polígono de la composición es un polígono válido según las restricciones establecidas para los polígonos.

- Polígonos no superpuestos: Los polígonos que forman la composición no se superponen. En otras palabras el área de intersección de los dos polígonos es cero.

- Área contigua: Cada polígono de la composición debe ser accesible desde cualquier otro polígono a través de la unión de los bordes del resto de polígonos, de este modo el área que forman es contigua.

#### 3.7.1.3.5 Sólido simple

Un sólido simple debe satisfacer las siguientes reglas de validación:

- Volumen contiguo: el volumen definido por el sólido debe ser contiguo.

- Cerrado: los límites del sólido son cerrados. Se comprueba validando que cada arista es atravesada dos veces en el sólido.

- Conectividad: Cada superficie que forma el sólido debe ser accesible a través de la unión de los bordes del resto de superficies que forman el sólido. Las aristas interiores deben tocarse para cumplir la condición de que el sólido sea cerrado, pero no pueden cruzarse.

- Superficies internas dentro de los límites: Cada superficie interna está dentro de los límites externos del sólido.

- Orientación apropiada: La superficie normal para todos los polígonos que forman el sólido tienen que apuntar hacía fuera del sólido.

- Superficies válidas: cada superficie que forma el sólido es una superficie válida.

- Polígonos sin anillos interiores: no se permite definir anillos interiores en la composición de superficies del sólido.

#### 3.7.1.3.6 Composición de sólidos

Una composición de sólidos debe satisfacer las siguientes reglas de validación:

- Sólidos simples válidos: cada sólido simple que forma la composición es un sólido simple válido, según las restricciones establecidas para los mismos.
- Volumen de intersección cero: El volumen de intersección de los sólidos que forman la composición debe ser cero.
- Conectividad: El volumen de la composición de los sólidos debe ser contiguo. Se puede pasar de cualquier punto de un componente a otro sin salir de la composición de sólidos.

En la siguiente figura se pueden observar ejemplos de geometrías no válidas según las restricciones indicadas para cada tipo de geometría en los apartados anteriores.


*Figura 63. Geometrías no válidas (Elaboración propia)*

## 3.7.1.4 Ejemplo de validación

A continuación, se van a presentar algunos ejemplos de uso de las funciones de validación vistas en el apartado anterior.

En primer lugar, se va a validar una única geometría haciendo uso de la función VALIDATE_GEOMETRY_WITH_CONTEX.

```
SELECT SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
(
  MDSYS.SDO_GEOMETRY
      (
2003,262154, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY (1,1003,1),
MDSYS.SDO_ORDINATE_ARRAY (12,13,14,11,18,11,20,13,16,16)
       ),
  0.25
)  resultado
FROM DUAL

RESULTADO
--------------------
13348 [Element <1>] [Ring <1>]
```

*Figura 64. Validación fallida usando VALIDATE_GEOMETRY_WITH_CONTEXT*

Consultado la documentación de Oracle, se puede comprobar que el error "ORA-13348" se produce porque el polígono indicado en el constructor SDO_GEOMETRY no se ha cerrado correctamente. Si se añaden las coordenadas que cierran el polígono y se ejecuta de nuevo la función se obtendrá el resultado correcto.

```
SELECT SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
(
  MDSYS.SDO_GEOMETRY
      (
2003,262154, NULL,
MDSYS.SDO_ELEM_INFO_ARRAY (1,1003,1),
MDSYS.SDO_ORDINATE_ARRAY (12,13,14,11,18,11,20,13,16,16,12,13)
       ),
  0.25
)  resultado
FROM DUAL

RESULTADO
--------------------
TRUE
```

*Figura 65. Validación correcta usando VALIDATE_GEOMETRY_WITH_CONTEXT*

En el ejemplo anterior se ha validado una geometría de forma individual, a continuación se va a ver un ejemplo de validación en la que se cargan todos los datos de la capa y posteriormente se validan todas las geometrías a la vez.

Supóngase que se ha cargado en la tabla 'LINEAS' el fichero que se muestra en la figura "*Figura 47. Ejemplo de carga con SQL Loader*". Se va a validar la capa, guardando los resultados de la validación en la tabla VALIDACION_RESULTADOS que previamente se habrá creado mediante el siguiente script:

```
CREATE TABLE VALIDACION_RESULTADOS(
  SDO_ROWID ROWID,
  STATUS VARCHAR2(2000));
```

*Figura 66. Tabla resultados VALIDATE_LAYER_WITH_CONTEXT*

Una vez creada la tabla, se lanza el script que realiza la validación e introduce los resultados en la tabla.

```
BEGIN
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT ('LINEAS', 'GEOMETRIA',
'VALIDACION_RESULTADOS');
COMMIT;
END;
/
```

*Figura 67. Ejemplo uso función VALIDATE_LAYER_WITH_CONTEXT*

Al ejecutar el script, si se detecta algún error, se registra en la tabla VALIDACION_RESULTADOS.

```
SELECT * FROM validacion_resultados;

SDO_ROWID                    STATUS
-----------------------------------  ------------------------------------------------------------
AAAHfvAABAAAMlqAAD           13356 [Element <1>] [Coordinate <2>]


SELECT cd_linea, descripcion, geometria FROM lineas
WHERE ROWID=' AAAHfvAABAAAMlqAAD';

CD_LINEA      DESCRIPCION          GEOMETRIA
---------------  ---------------------  ------------------------------------------------------------------------
---------4                  l4                    SDO_GEOMETRY(2002, 262144, NULL,
                                                  SDO_ELEM_INFO_ARRAY(1, 2, 1),
                                                  SDO_ORDINATE_ARRAY(3, 6, 7, 1, 7,
                                                 1))
```

*Figura 68. Resultado validación VALIDATE_LAYER_WITH_CONTEXT*

Observando la figura anterior se puede comprobar que la línea con código ‘4’ no está bien definida, dado que se repite un vértice en sus coordenadas, eso es lo que indica el error “ORA-13356” de la referencia de Oracle.

## 3.7.2 Depuración

Uno de los errores vistos en el apartado anterior es debido a la duplicidad de los vértices de las geometrías. Además de las funciones de validación Oracle proporciona en el paquete SDO_UTIL algunas funciones destinadas a depurar y limpiar datos, como por ejemplo la utilizada para eliminar vértices duplicados en geometrías. A continuación se presentan algunas de ellas. Se pueden consultar todas las disponibles en la referencia de Oracle Spatial.

### 3.7.2.1 REMOVE_DUPLICATE_VERTICES

Utilizando esta función se eliminan los vértices duplicados en una geometría. Recibe como parámetro el objeto SDO_GEOMETRY a depurar y el valor de la tolerancia a aplicar en la búsqueda de vértices duplicados, y devuelve otro objeto SDO_GEOMETRY en el que se han eliminado los vértices duplicados.

Basado en el error de validación de la figura "*Figura 68. Resultado validación VALIDATE_LAYER_WITH_CONTEXT*" se va a hacer uso de la función REMOTE_DUPLICATE_VERTICES para obtener una geometría valida en la que no se repitan los vértices:

```
SELECT geometria,
        SDO_UTIL.REMOVE_DUPLICATE_VERTICES(geometria,0.25) nodup_geometria
FROM lineas
WHERE cd_linea=4;

GEOMETRIA                                          NODUP_GEOMETRIA
-------------------------------------------------- --------------------------------------------------
---
SDO_GEOMETRY(2002, 262144, NULL,                   SDO_GEOMETRY(2002, 262144, NULL,
      SDO_ELEM_INFO_ARRAY(1, 2, 1),                      SDO_ELEM_INFO_ARRAY(1, 2, 1),
      SDO_ORDINATE_ARRAY(3, 6, 7, 1, 7, 1))              SDO_ORDINATE_ARRAY(3, 6, 7, 1))
```

*Figura 69. Ejemplo uso función REMOVE_DUPLICATE_VERTICES*

### 3.7.2.2 EXTRACT

Sirve para extraer un elemento específico de un objeto SDO_GEOMETRY. Puede ser de mucha utilidad en la depuración de colecciones como multipolígonos. Recibe como parámetros el objeto SDO_GEOMETRY del que extraer el elemento, el número de elemento a extraer y opcionalmente el número de anillo del elemento. Devuelve el elemento extraído como un objeto SDO_GEOMETRY.

A continuación se muestra un ejemplo de extracción de un polígono de un multipolígono.

```
SELECT SDO_UTIL.EXTRACT
(
        SDO_GEOMETRY
        (
                2007, -- Colección de polígonos
                NULL,
                NULL,
                SDO_ELEM_INFO_ARRAY
                (
                        1, 1003, 3,  -- primer elemento
                        5, 1003, 1 -- segundo elemento
                ),
```

```
                SDO_ORDINATE_ARRAY
                (
                        1, 1, 2, 2, -- ordenadas primer elemento
                        3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3 -- ordenadas segundo elemento
                )
        ),-- Final de la geometría de la que se extrae el elemento
        2 -- Extracción del segundo elemento
)  segundo_elemento
FROM dual;

SEGUNDO_ELEMENTO
--------------------------------------------------------------------------------
SDO_GEOMETRY
(
        2003,
        NULL,
        NULL,
        SDO_ELEM_INFO_ARRAY(1, 1003, 1),
        SDO_ORDINATE_AR RAY
        (
                3, 3,     -- valores coordenadas primer punto
                4, 3,     -- valores coordenadas segundo punto
                4, 4,     -- valores coordenadas tercer punto
                3, 4,     -- valores coordenadas cuarto punto
                3, 4,     -- valores coordenadas quinto punto
                3, 3      -- valores coordenadas sexto punto
        )
)
```

*Figura 70. Ejemplo de uso de la función EXTRACT*

Obsérvese que el elemento que se obtiene es el segundo del SDO_ELEM_INFO_ARRAY (5, 1003, 1), que empieza en la quinta coordenada, es decir, el tercer punto.

# 3.8 Geocoder

En el apartado anterior, se ha podido observar el funcionamiento de algunos de los mecanismos disponibles para cargar datos en las tablas desde fuentes de datos externas.

El ‘Geocoder’ es un componente específico de Oracle Spatial, que sirve para realizar el proceso de carga, transformando información espacial implícita, en objetos de tipo SDO_GEOMETRY.

A lo largo de este apartado, se va a describir en qué consiste el proceso de geocodificación. En los ejemplos que se verán, cada una de la entidades que se manipulan (clientes, almacenes, etc.) son localizables espacialmente. Todas ellas incluyen una columna del tipo SDO_GEOMETRY, que contiene su localización espacial, usando las coordenadas geográficas (longitud y latitud).

Esta forma de trabajo, plantea la primera pregunta. ¿De dónde viene esta información? Ciertamente, no es posible pedir a los clientes sus coordenadas geográficas una vez se han decidido por el producto, sino que lo más frecuente es lanzar un proceso de geocodificación (geocoding). Esto no es más que geocodificar sus direcciones y almacenar el resultado de la localización como un objeto SDO_GEOMETRY.

Se debe entender por "geocodificar", el proceso que convierte las direcciones postales (por ejemplo, Río Duero 71, Leganés, Madrid, 28913, España), en coordenadas geográficas (longitud = -77.060283, latitud= 38.9387083).

Además, el proceso de geocodificación, puede normalizar y validar la entrada de direcciones (Número=71; Calle=Río Duero; Localidad=Leganés; Provincia=Madrid;C.P.=28913; País=España).

Inicialmente se va a describir la funcionalidad del Geocoder y cómo se puede usar la localización en un entorno de trabajo.

Se comenzará con una introducción al proceso de geocodificación que permita entender cómo se realiza la conversión entre direcciones a objetos SDO_GEOMETRY.

Seguidamente se darán más detalles sobre cómo conocer más sobre la manera de localizar y extrapolar la localización de una dirección específica. Y por último se describirán brevemente las diferentes funciones de geocodificación.

## 3.8.1 Proceso de geocodificación

El proceso de geocodificación tiene dos propósitos. El primero y principal de ellos es asociar coordenadas geográficas con direcciones. Por ejemplo usando la función 'GEOCODE_AS_GEOMETRY', que devuelve un puntero a un objeto 'SDO_GEOMETRY', obtendríamos las coordenadas geométricas que el Geocoder ha asociado a la dirección.

Una vez llegados a este punto podremos profundizar en el segundo propósito del Geocoder, que no es otro que la validación y corrección de errores en las entradas de direcciones. A este proceso se le denomina normalización e implica la corrección de errores, asegurando que toda la información de direcciones es completa, bien estructurada y clara. Es la única manera de asegurar que los datos devueltos son legibles y evitar la introducción de duplicados.

Gracias a este proceso a través de la función 'SDO_GEO_ADR', somos capaces de devolver una dirección correcta, a partir de una dirección introducida incorrectamente, debido, por ejemplo a errores sintácticos a la hora de escribir la calle o códigos postales incorrectos.

## 3.8.2 Arquitectura del geocoder de Oracle

Primeramente se necesita introducir una lista de direcciones con coordenadas conocidas, como pueden ser carreteras, calles, ciudades, códigos postales, así como su localización geográfica y formas.

Con este conjunto de datos inicial, el Geocoder realizar los tres pasos siguientes:

- Analiza la dirección de entrada

- Busca por nombre entre las direcciones existentes

- Calcula la localización (coordenadas espaciales) para la dirección encontrada.

Para realizar el proceso de análisis, el Geocoder divide la dirección de entrada en elementos con entidad, tal como, el nombre de la calle, el tipo, el número, el código postal y la ciudad.

Este proceso es complejo dado que, por ejemplo, existen diferentes maneras de expresar una dirección dependiendo del país. El Geocoder de Oracle reconoce una gran variedad de formatos de direcciones según varios países y lenguajes.

Estos formatos están definidos en la tabla, GC_PARSER_PROFILEAFS.

Una vez que la dirección ha sido analizada y separada en elementos, el Geocoder busca en la lista de nombres de calles una que coincida, lo máximo posible, con la dirección dada.

Esta búsqueda es confusa, ya que puede devolver valores incluso si la dirección esta introducida de forma incorrecta, o esta almacenada de otra manera por ejemplo (calle vs. C. o Street vs St.), las diferentes palabras claves que se han usado en la dirección, (con sus diferentes formas de ser interpretadas, al igual que los errores sintácticos más comunes), son almacenados en la tabla GC_PARSER_PROFILES.

Esta búsqueda puede ser por proximidad, esto es, si el nombre de la calle no se ha encontrado, el Geocoder volverá a realizarla por el código postal o por el nombre de la ciudad. De esta manera se obtendrán múltiples coincidencias. En este momento se debe decidir qué resultado elegir, si se deja a la aplicación que elija el resultado que debe devolver o, más comúnmente, se deja al usuario de la aplicación que elija el resultado correcto.

Por último, tras realizar la búsqueda de la dirección, el Geocoder debe transformarla en un punto geográfico.

Los datos de referencia del Geocoder usados permiten establecer el número de una casa a cada segmento de una calle. Cuando una dirección se introduce con el número, el Geocoder es capaz de calcular la posición geográfica del número introducido mediante interpolación.

El geocoder asume que las casas se encuentran con una separación regular, a lo largo de la línea geométrica que representa el segmento de calle, por lo que es capaz de calcular, de manera aproximada la localización exacta, o con margen aceptable de error, del número introducido.

Esta forma de calcular la posición tiene una serie de matices:

- En el caso que la dirección no tenga información del número, el geocoder devuelve el punto intermedio, del segmento de la calle, y guarda este cálculo del número.

- Si el nombre de la calle no se ha dado en la dirección o simplemente no se ha encontrado, el geocoder realizará una búsqueda por código postal, o ciudad, etc.…En estos casos devolverá el punto que se corresponda con el centro de elemento de búsqueda, ya sea la ciudad, el “centro” del código postal.

## 3.8.3 Creación de los datos de referencia del Geocoder

Los datos de referencia usados por el geocoder de Oracle, son un conjunto de tablas con una estructura específica. Todas las tablas empiezan con el prefijo 'GC_'.

Existen dos tipos de tablas:

- Tablas de parámetros: controlan las operaciones del geocoder
- Tablas de datos: Contienen los nombres de los lugares y sus coordinadas geográficas.

La manera de rellenar estas tablas dependerá del suministrador de los datos. Por ejemplo NAVTEQ, suministra los datos de referencia para el geocoder en ficheros (.dmp) o tablespaces transportables. Otros mecanismos pudieran ser scripts SQL.

### 3.8.3.1 Tablas de parámetros

Tres tablas contienen información acerca de las estructuras de las direcciones soportadas en cada país por el geocoder de Oracle. El contenido de estas tablas no debería modificarse.

- GC_COUNTRY_PROFILE: Esta tabla contiene la información general, sobre cada país conocido por el geocoder de Oracle. Uno de los datos importantes es el sufijo de las tablas de datos para cada país. (se entrará en detalle a continuación).
- GC_PARSER_PROFILEAFS: En esta tabla se describen las estructuras de las direcciones para cada país soportado por el geocoder de Oracle. Existe una fila por cada país, con la estructura de la dirección definida en notación XML.
- GC_PARSER_PROFILES: El geocoder de Oracle usa esta tabla para reconocer algunos de los elementos de las direcciones. Se definen los elementos de las direcciones junto con sus sinónimos, incluyendo los errores de sintaxis más frecuentes. Por ejemplo, se define que AV, AVE, AVEN, AVENI, AVN y AVENDA, son todas las maneras posibles de escribir AVENIDA.

### 3.8.3.2 Tablas de datos

Los nombres de las tablas de datos están formados por el sufijo específico de cada país, (definido en la tabla GCL_COUNTRY_PROFILE). Por ejemplo los datos referentes a Francia, en estas tablas vienen definidos por el sufijo 'FR', mientras que para Estados Unidos se usa el sufijo 'US'. En las siguientes descripciones de tablas las "xx" representan este sufijo.

- GC_AREA_xx: esta tabla almacena información de todas las áreas administrativas. El geocoder de Oracle define tres niveles de áreas administrativas: REGION, MUNICIPALITY, SETTLEMENT, la forma de

definir estos niveles dependerá del país en el que nos encontrados, ya que en España sería, Provincia, Localidad y Municipio, mientras que en Estados Unidos sería Estados, condados y ciudades.

- GC_POSTAL_CODE_xx: en esta tabla se describen todos los códigos postales, esto incluye los coordinadas del "punto central" para cada código postal. El punto central es devuelto por el geocoder cuando el nombre de la calle introducida es inválido, o no se ha introducido.

- GC_POI_xx: esta Tabla contiene una selección de los puntos de interés, (hospitales, aeropuertos, hoteles, restaurantes, parkings, etc..), El número de puntos de interés (POIs) y su clasificación variará dependiendo del suministrador de datos.

- GC_ROAD_xx: esta es la tabla principal para realizar las búsquedas de direcciones. Contiene una fila por cada carretera, localidad y código postal. Si la carretera cruza varios códigos postales, puede aparecer múltiples veces en esta tabla.

- GC_ROAD_SEGMENT_xx: esta tabla obtiene la información necesaria para calcular las coordenadas de una dirección mediante interpolación. Contiene una fila por cada segmento de carreteras con la forma geométrica del propio segmento (de tipo SDO_GEOMETRY), así como los números de casas para cada lado, al igual que al final del segmento.

- GC_INTERSECTION_xx: cuando múltiples segmentos de carreteras se encuentran, forman una intersección. En esta tabla se encuentra una fila por cada pareja de segmentos.

## 3.8.4 Funciones del Geocoder

La API del geocoder es muy simple, está compuesta por un paquete PL/SQL (SDO_GCDR), con sólo unas pocas funciones. Todas ellas aceptan la introducción de direcciones y devuelven las coordenadas geográficas correspondientes. La diferencias entre cada una de ellas es la cantidad de información que devuelven.

| Función | Conversión direcciones | Corrección direcciones | Descripción |
|---|---|---|---|
| GEOCODE_AS_GEOMETRY | SI | NO | Devuelve un punto geométrico (con sus coordenadas geográficas) para la dirección introducida. No devuelve información sobre la precisión y calidad del resultado. Esta función es la mejor cuando sabemos que la dirección introducida es correcta. |

| Función | Conversión direcciones | Corrección direcciones | Descripción |
|---|---|---|---|
| GEOCODE | SI | SI | Devuelve las coordenadas geográficas y corrige la dirección con indicaciones detalladas sobre la calidad del resultado. El resultado lo devuelve en el tipo de objeto SDO_GEO_ADDR. |
| GEOCODE_ADDR | SI | SI | Igual que GEOCODE, pero como entrada de los datos a geocodificar también usa un objeto de tipo SDO_GEO_ADDR. |
| GEOCODE_ALL | SI | SI | Igual que GEOCODE, pero puede devolver múltiples coincidencias si la dirección de entrada es ambigua. Esta función es utilizada en la mayoría de los casos cuando es el usuario final el que decide cual es el resultado correcto. El resultado lo devuelve en el tipo de objeto SDO_ADDR_ARRAY. |
| GEOCODE_ALL_ADDR | SI | SI | Igual que GEOCODE_ALL, pero como entrada de los datos a geocodificar también usa un objeto de tipo SDO_GEO_ADDR. |

*Tabla 7. Funciones del Geocoder*

# 3.9 Índices espaciales

En los apartados "*3.6 Carga de datos espaciales*" y "*3.7 Validación y depuración*" se han explicado los procesos de carga y validación de datos. Como ya se ha mencionado en esos puntos, es recomendable realizar la carga de datos antes de definir los índices, e incluso borrar los ya existentes y una vez finalizada la carga volver a incluirlos.

Un índice espacial, como cualquier otro índice, proporciona un mecanismo para limitar la búsqueda y así optimizar las consultas, pero en este caso basado en criterios espaciales tales como intersección y contención.

En este apartado se va a definir en qué consiste un índice espacial y la sintaxis de creación de los mismos en Oracle Spatial.

## 3.9.1 Conceptos de indexación espacial

Cuando creamos un índice espacial, internamente Oracle lo implementa por defecto como un índice R-tree. También es posible crear índices de tipo QuadTree, aunque no se recomienda dado que solo pueden indexar hasta dos dimensiones y requieren un ajuste explícito.

Un índice R-tree aproxima cada geometría a un solo rectángulo (llamado el rectángulo de limitación mínimo, o MBR), que incluye como mínimo la geometría.

Para una capa de geometrías un R-tree consiste en una estructura jerárquica de MBRs correspondientes a las geometrías de la capa.

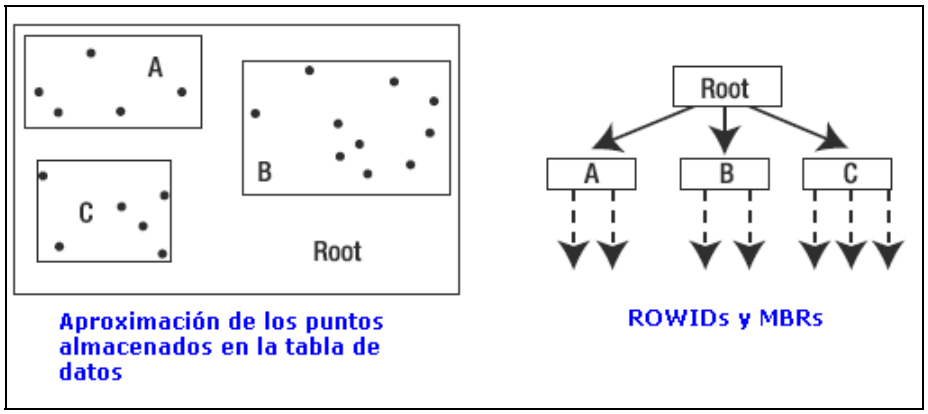



*Figura 71. Ejemplo de R-tree [KGE, 2007]*

Suponiendo que los puntos de la figura anterior correspondan a puntos almacenados en la tabla 'PUNTOS' del escenario de pruebas, para cada punto, el R-tree calcula el rectángulo mínimo circunscrito (MBR) que lo incluye y crea una jerarquía de MBRs. Por ejemplo, en la figura anterior, la ubicación de los puntos se agrupa en los tres nodos A B y C; y cada nodo se asocia con un MBR que incluye la ubicación de los datos en otros sub-árboles. De modo que el R-tree construye una estructura de árbol jerárquico con los MBRs de los puntos almacenados en la columna 'GEOMETRIA' de la tabla 'PUNTOS'. A continuación, utiliza esta jerarquía de MBRs para orientar las consultas a las ramas adecuadas del árbol y finalmente a las filas de la tabla 'PUNTOS'.

La figura "*Figura 72. Almacenamiento índice R-tree*" muestra cómo almacena Oracle el índice R-tree.

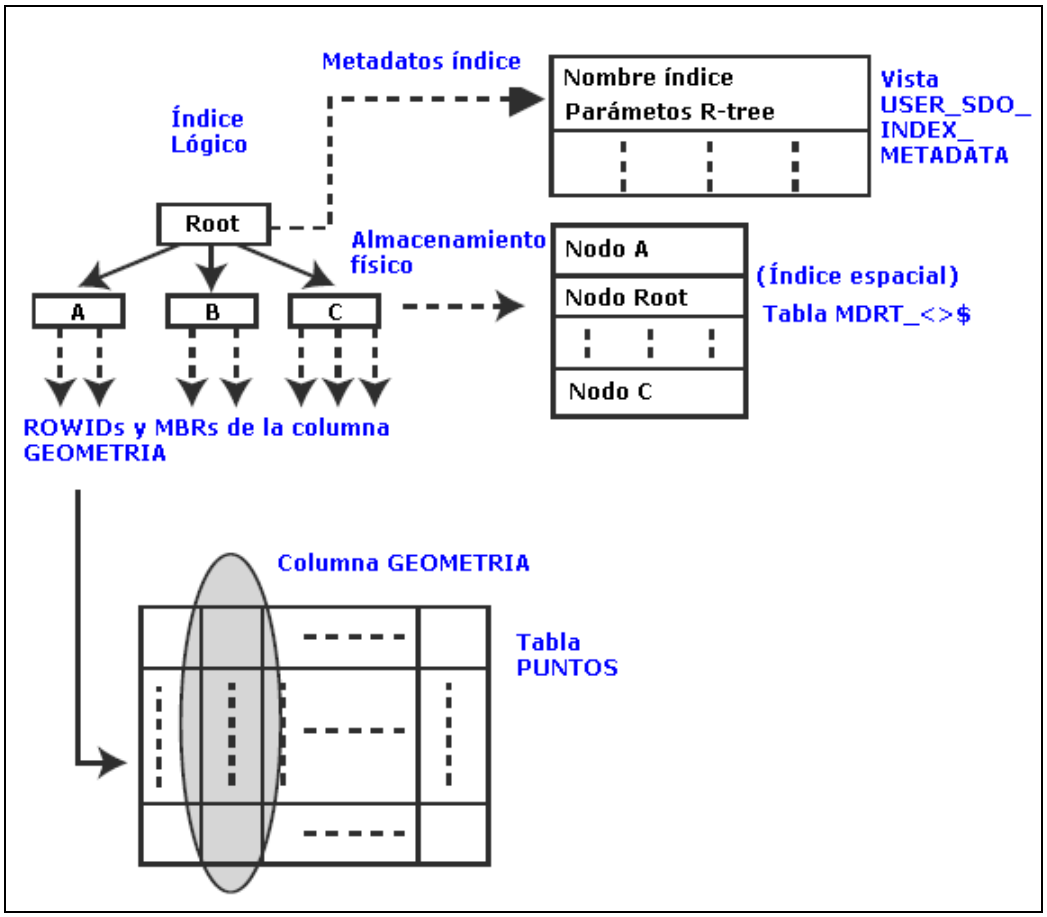



*Figura 72. Almacenamiento índice R-tree [KGE, 2007]*

La estructura de árbol lógico se almacena como una tabla de Oracle que comienza con el prefijo 'MDRT'. Cada nodo de la estructura del árbol se almacena como una fila de la tabla. Los metadatos del índice se encuentran en la vista USER_SDO_INDEX_METADATA. Se almacena el nombre del índice espacial, la tabla que contiene el índice, el ROWID del root del índice R-tree y otros parámetros relevantes que como se verá más adelante, se indican en la sentencia de creación del índice.

El estado de la estructura del índice es muy importante. Las operaciones de borrado e inserción pueden llegar a afectar significativamente al árbol R-tree, degradando la calidad de su estructura y por tanto afectando al rendimiento de las consultas. Oracle provee algunas funciones y procedimientos relativos a la calidad de los índices, que pueden utilizarse para mantener los índices en buen estado, tales como, SDO_TUNE.QUALITY_DEGRADATION, SDO_TUNE.ANALYZE_RTREE o SDO_TUNE.TREE_QUALITY.

## 3.9.2 Sintaxis de creación

Una vez comprendidos los conceptos básicos de indexación espacial, se va a ver la sintaxis a utilizar para crear un índice espacial. Es la siguiente:

```
CREATE INDEX <nombre_indice> ON <tabla>(<columna>)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
[PARAMETERS ('parametros')];
```

*Figura 73. Sintaxis de creación de un índice espacial*

Donde:

- nombre_indice: es el nombre a asignar al índice.
- tabla: determina el nombre de la tabla sobre la que se va a crear el índice.
- columna: nombre de la columna que contiene los datos a indexar.
- parámetros: lista de parámetros que determinan características propias del índice. Se indican expresados en modo nombre_parametro=valor_parametro.

A continuación se presentan algunos ejemplos de creación de índices, a la vez que se describen los parámetros más utilizados en su creación.

### 3.9.2.1 Parámetro TABLESPACE

Indica en que tablespace se quiere almacenar el índice espacial.

```
CREATE INDEX puntos_spatial_idx ON puntos(geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('TABLESPACE=MI_TABLESPACE_IDX);
```

*Figura 74. Creación del índice con el parámetro TABLESPACE*

### 3.9.2.2 Parámetro WORK_TABLESPACE

Durante la creación del índice, el índice R-tree realiza operaciones de ordenación en el conjunto de datos. Para ello, crea algunas tablas de trabajo que se eliminan al final de la creación del índice. La creación y borrado de muchas tablas de diferentes tamaños puede fragmentar el espacio del tablespace. El parámetro WORK_TABLESPACE permite definir un tablespace de trabajo a parte, evitando que los tablespaces de índices o datos se fragmenten al realizar las operaciones de creación del índice.

Si no se establece, las tablas de trabajo se crean en el mismo tablespace de creación del índice.

```
CREATE INDEX puntos_spatial_idx ON puntos(geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('WORK_TABLESPACE=MI_TABLESPACE_AUX);
```

*Figura 75. Creación del índice con el parámetro WORK_TABLESPACE*

El tamaño total (en bytes) usado en el tablespace de trabajo será de aproximadamente 200 a 300 veces el número de filas en la tabla de ‘PUNTOS’.

Es importante señalar que las tablas de trabajo que se crean no son tablas temporales por lo que no se debe usar el tablespace temporal como tablespace de trabajo.

### 3.9.2.3 Parámetro LAYER_GTYPE

Este parámetro sirve para indicar el tipo específico de geometría que se almacena en la columna sobre la que se está creando el índice, por defecto todos los tipos están permitidos. Esto ayudará en la comprobación de la integridad y en ocasiones en acelerar el resultado de los operadores de consulta.

```
CREATE INDEX puntos_spatial_idx ON puntos(geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS (' LAYER_GTYPE=POINT);
```

*Figura 76. Creación del índice con el parámetro LAYER_GTYPE*

En la figura anterior se establece el valor del parámetro a POINT para indicar que la columna 'GEOMETRIA' de la tabla 'PUNTOS', sólo almacena geometrías de tipo punto. Al haber establecido el tipo si alguien tratase de insertar en la tabla otro tipo de geometría distinto Oracle lanzaría un error no permitiendo la inserción.

En general se puede establecer el valor a los nombres de los SDO_GTYPEs (POINT, LINE, POLYGON, etc).

### 3.9.2.4 Parámetro SDO_INDX_DIMS

Indica el número de dimensiones a indexar, por defecto se establece el valor 2, pero pueden indexarse hasta 4 dimensiones.

```
CREATE INDEX puntos_spatial_idx ON puntos(geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS (' SDO_INDX_DIMS=2);
```

*Figura 77. Creación del índice con el parámetro SDO_INDX_DIMS*

### 3.9.2.5 Parámetro SDO_DML_BATCH_SIZE

Cuando se añaden, modifican o eliminan registros de una tabla que contiene un índice espacial, los cambios en el índice no se actualizan automáticamente, sino que se realizan por bloques. Este parámetro sirve para establecer el tamaño del lote de inserciones, borrados y modificaciones en una transacción. Si no se especifica por defecto toma el valor 1000, es decir, las actualizaciones son incorporadas al índice en bloques de 1000. Es un valor recomendado para la mayoría de las transacciones que combinan una mezcla de operaciones de consulta, inserción, borrado y modificación. Sin embargo si las transacciones contienen un número muy elevado de inserciones, modificaciones y borrados, es recomendable establecer el valor entre 5000 y 10000. Esto mejorará el rendimiento del commit de la transacción, pero a la vez consumirá más memoria y más recursos del sistema. En general el valor se puede establecer entre 1 y 10000.

```
CREATE INDEX puntos_spatial_idx ON puntos(geometria)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('SDO_DML_BATCH_SIZE=1000);
```

*Figura 78. Creación de índice con el parámetro SDO_DML_BATCH_SIZE*

Tal y como se ha definido el tamaño de lote en la figura anterior, si por ejemplo se insertaran 3500 filas en la tabla 'PUNTOS' y se realizara el commit de la transacción, las actualizaciones en la tabla del índice espacial 'PUNTOS_SPATIAL_IDX' se llevarían a cabo en cuatro lotes de la operaciones de inserción (1000, 1000, 1000, 500).

# 3.10 Análisis espacial

En el apartado anterior se ha visto la creación de índices en las tablas como mecanismo para limitar las búsquedas sobre la información espacial.

En este apartado se va a presentar el modelo de consulta utilizado por Spatial para analizar las búsquedas sobre la información espacial y algunos de los operadores y funciones más utilizados en las consultas.

## 3.10.1 Escenario de trabajo de consultas

En las tablas 'PUNTOS', 'LINEAS' y 'POLIGONOS' se va a cargar el escenario que se muestra en la siguiente figura:

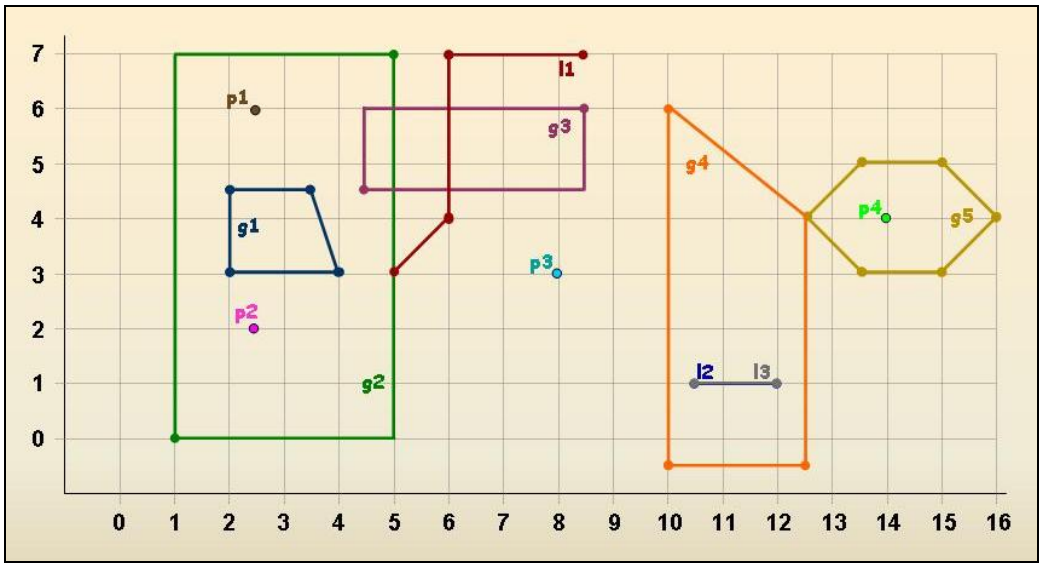



*Figura 79. Escenario de trabajo de operadores y funciones (Elaboración propia)*

Está compuesto por las geometrías:

| Geometría | Coordenadas |
|---|---|
| g1 | {2 \| 3 \| 4 \| 3 \| 3.5 \| 4.5 \| 2 \| 4.5 \| 2 \| 3} |
| g2 | {1 \| 1 \| 5 \| 7} |
| g3 | {4.5 \| 4.5 \| 8.5 \| 6} |
| g4 | {10 \| 0.5 \| 12.5 \| 0.5 \| 12.5 \| 4 \| 10 \| 6 \| 10 \| 0.5} |
| g5 | {12.5 \| 4 \| 13.5 \| 3 \| 15 \| 3 \| 16 \| 4 \| 15 \| 5 \| 13.5 \| 5 \| 12.5 \| 4} |
| l1 | {5 \| 3 \| 6 \| 4 \| 6 \| 7 \| 8.5 \| 7} |
| l2 | {10.5 \| 1.5 \| 12 \| 1.5} |
| l3 | {10.5 \| 1.5 \| 12 \| 1.5} |
| p1 | {2.5 \| 6} |
| p2 | {2.5 \| 2} |
| p3 | {8 \| 3} |
| p4 | {14 \| 4} |

*Tabla 8. Geometrías de trabajo de operadores y consultas*

Las tablas se han cargado mediante SQL Loader con los ficheros de carga que se muestran a continuación.

```
LOAD DATA
INFILE *
INTO TABLE puntos
FIELDS TERMINATED BY '|'
TRAILING NULLCOLS (
cd_punto INTEGER EXTERNAL,
descripcion CHAR,
geometria COLUMN OBJECT
(
        SDO_GTYPE INTEGER EXTERNAL,
        SDO_SRID INTEGER EXTERNAL,
        SDO_POINT COLUMN OBJECT
                (X FLOAT EXTERNAL,
                 Y FLOAT EXTERNAL)
)
)

BEGINDATA
1|p1|2001|262144|2,5|6|
2|p2|2001|262144|2,5|2|
3|p3|2001|262144|8|3|
4|p4|2001|262144|14|4|
```

*Figura 80. Fichero carga escenario puntos.ctl*

```
LOAD DATA
INFILE *
CONTINUEIF NEXT(1:1)='#'
INTO TABLE lineas
FIELDS TERMINATED BY '|'
TRAILING NULLCOLS (
cd_linea INTEGER EXTERNAL,
descripcion CHAR,
geometria COLUMN OBJECT
(
        SDO_GTYPE INTEGER EXTERNAL,
        SDO_SRID INTEGER EXTERNAL,
        SDO_ELEM_INFO VARRAY terminated by '/' (elementos FLOAT EXTERNAL),
        SDO_ORDINATES VARRAY terminated by '/' (ordenadas FLOAT EXTERNAL)

)
}
BEGINDATA
 1|Linea 1|2002|262144|
#1|2|1|/
#5|3|6|4|6|7|8,5|7|/
 2|Linea 2|2002|262144|
#1|2|1|/
#10,5|1,5|12|1,5|/
 3|Linea 3|2002|262144|
#1|2|1|/
#10,5|1,5|12|1,5|/
```

*Figura 81. Fichero carga escenario lineas.ctl*

```
LOAD DATA
INFILE *
CONTINUEIF NEXT(1:1)='#'
INTO TABLE poligonos
FIELDS TERMINATED BY '|'
TRAILING NULLCOLS (
cd_poligono INTEGER EXTERNAL,
descripcion CHAR,
geometria COLUMN OBJECT
(
        SDO_GTYPE INTEGER EXTERNAL,
        SDO_SRID INTEGER EXTERNAL,
        SDO_ELEM_INFO VARRAY terminated by '/' (elementos FLOAT EXTERNAL),
        SDO_ORDINATES VARRAY terminated by '/' (ordenadas FLOAT EXTERNAL)

)
)

BEGINDATA
 1|g1|2003|262144|
#1|1003|1|/
#2|3|4|3|3,5|4,5|2|4,5|2|3|/
 2|g2|2003|262144|
#1|1003|3|/
#1|1|5|7|/
 3|g3|2003|262144|
#1|1003|3|/
#4,5|4,5|8,5|6|/
 4|g4|2003|262144|
#1|1003|1|/
#10|0,5|12,5|0,5|12,5|4|10|6|10|0,5|/
 5|g5|2003|262144|
#1|1003|1|/
#12,5|4|13,5|3|15|3|16|4|15|5|13,5|5|12,5|4|/
```

*Figura 82. Fichero carga escenario poligonos.ctl*

Los metadatos definidos para cada tabla son los que se muestran en la figura "*Figura 46. Metadatos capas puntos, líneas y polígonos*".

Se ha utilizado el sistema de referencia local con SRID "262144", sistema en dos dimensiones X e Y, unidad de medida es el centímetro.

Es importare recordar que tras cargar los datos se deben crear de nuevo los índices espaciales sobre las columnas SDO_GEOMETRY de las tablas.

Las geometrías de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*" van a constituir el escenario de trabajo sobre el que aplicar los operadores y funciones incluidos en los siguientes apartados. Se ha elegido un escenario sencillo de modo que sea fácil verificar el resultado de las consultas.

## 3.10.2 Operadores

Los operadores están ligados al índice espacial y se evalúan haciendo uso de un modelo en dos niveles. El término dos niveles, indica que se realizan dos operaciones distintas, lo que se denominan filtros primario y secundario.

En la siguiente figura se puede observar la representación de modelo:

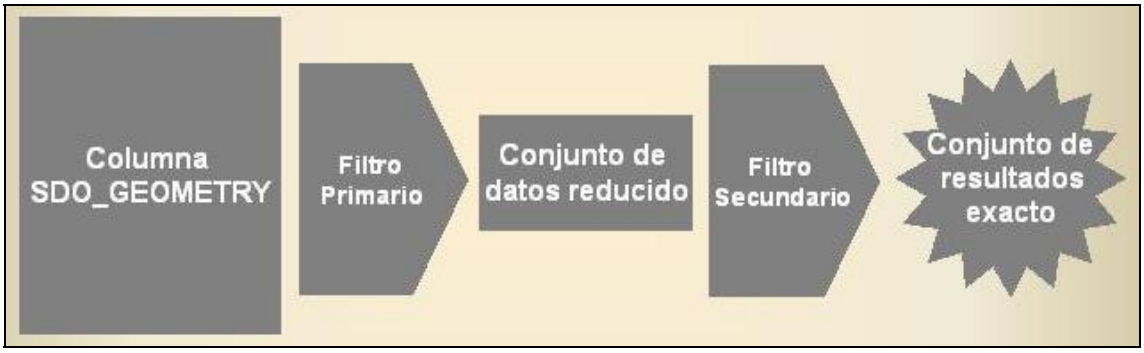



*Figura 83. Modelo de consulta de Spatial [MAB+, 2009]*

El filtro primario, permite una rápida selección de registros candidatos para pasar el segundo filtro que dará el conjunto de resultados exacto. Compara aproximaciones geométricas para reducir la complejidad de computación, por tanto se considera un filtro poco costoso y más rápido. Las aproximaciones del índice (los MBRs almacenados en la tabla del índice espacial) se utilizan para identificar un conjunto de filas candidatas a cumplir la relación establecida por el operador en la consulta.

A veces, este primer filtro es suficiente para obtener el resultado buscado. Por ejemplo una ampliación en un mapa para un área determinada, se puede obtener recuperando la interacción de las geometrías con un rectángulo.

El filtro secundario, aplica computaciones exactas, sobre los datos que se obtienen tras haber aplicado el filtro primario. La operación es más costosa por ello se aplica solo sobre el subconjunto de datos obtenidos por el filtro primario.

Todo el proceso es transparente para el usuario, simplemente utiliza el operador en la cláusula WHERE de la sentencia SQL e internamente se aplican los filtros necesarios para obtener el número correcto de filas.

### 3.10.2.1 Sintaxis genérica de un operador

La sintaxis genérica de un operador es la siguiente:

```
<operador>
(
        table_geometry IN SDO_GEOMETRY,
        query_geometry IN SDO_GEOMETRY
        [, parameter_string IN VARCHAR2
        [, tag IN NUMBER ]]
) = 'TRUE'
```

*Figura 84. Sintaxis genérica de un operador*

Donde:

- table_geometry: es el nombre de la columna SDO_GEOMETRY de la tabla sobre la que se va a aplicar el operador. Debe tratarse de una columna indexada espacialmente, de lo contrario el operador lanzará un error.

- query_geometry: es la consulta de localización. Puede tratarse de una columna SDO_GEOMETRY de otra tabla, de una variable de enlace o de un objeto SDO_GEOMETRY construido dinámicamente.

- parameter_string: son los parámetros específicos del operador. El símbolo corchete indica que son opcionales en algunos operadores. Se indican expresados en modo nombre_parametro=valor_parametro.

- tag: especifica un número que se utiliza sólo en determinados operadores espaciales. De nuevo, el símbolo corchete indica que es un parámetro opcional. Y sólo puede especificarse en conjunción con el argumento de "parameter_string".

Tal y como se observa en la figura "*Figura 84. Sintaxis genérica de un operador*" Oracle Spatial estipula que un operador se equipara siempre a la cadena 'TRUE'. Esto lo convierte en un predicado a ser evaluado con respecto a cada fila de la tabla indexada.

En los siguientes apartados se describen algunos de los operadores de uso más común, su sintaxis específica y algunos ejemplos concretos. Tal y como se verá en estos ejemplos, cabe destacar que los operadores solo pueden ser utilizados formando parte de la cláusula 'WHERE' de la consulta.

### 3.10.2.2 Operador SDO_FILTER

El operador SDO_FILTER identifica todas las filas de una tabla donde los MBRs de las geometrías de la columna indexada, tienen intersección con el MBR de la geometría de consulta. Ejecuta únicamente el filtro primario por lo que es el operador más rápido. En general sirve de base a otros operadores basados en la intersección, ya que suele devolver más candidatos de los que realmente se cruzan con la geometría de consulta. Aunque dado que el margen de error es pequeño, en sistemas tales como visualizadores de mapas, en los que se puede admitir dicho margen, puede ser muy útil dada su rapidez frente al resto de operadores.

Por otro lado es el único operador que puede usarse para índices construidos en más de dos dimensiones.

La sintaxis del operador SDO_FILTER es la siguiente:

```
SDO_FILTER
(
        table_geometry IN SDO_GEOMETRY,
        query_geometry IN SDO_GEOMETRY
        [, parameter_string IN VARCHAR2 ]
) = 'TRUE'
```

*Figura 85. Sintaxis operador SDO_FILTER*

Los parámetros 'table_geometry' y 'query_geometry' se encuentran descritos en el apartado "*3.10.2.1 Sintaxis genérica de un operador*". Los parámetros específicos que se pueden indicar en 'parameter_string' son:

- querytype=<valor cadena>: establecido siempre como ‘querytype=window’. Es opcional a partir de la versión 10g, sin embargo en versiones anteriores es obligatorio indicarlo.

- min_resolution=<valor numérico>: parámetro opcional que sirve para incluir geometrías donde al menos uno de los lados del MBR de la geometría es igual o mayor que el valor especificado. Por ejemplo, “min_resolution=10” incluye solamente las geometrías para las que la anchura, la altura o ambos del MBR de la geometría es al menos 10. Sirve para excluir geometrías que son demasiado pequeñas para ser de interés en la consulta.

- max_resolution=<valor numérico>: parámetro opcional que sirve para incluir geometrías donde al menos uno de los lados del MBR de la geometría es igual o menor que el valor especificado. Por ejemplo, ‘min_resolution=10’ incluye solamente las geometrías para las que la anchura, la altura o ambos del MBR de la geometría es menor o igual que 10. Sirve para excluir geometrías que son demasiado grandes para ser de interés en la consulta.

Como ejemplo, basado en el escenario de la figura “*Figura 79. Escenario de trabajo de operadores y funciones*” se van a recuperar los polígonos que tienen intersección con la línea con código igual a ‘1’.

```
SELECT pl.cd_poligono, pl.descripcion
FROM poligonos pl, lineas l
WHERE l.cd_linea = 1AND
        MDSYS.SDO_FILTER (pl.geometria, l.geometria, 'querytype=WINDOW') = 'TRUE'

CD_POLIGONO          DESCRIPCION
-------------------------  ----------------------
2                    g2
3                    g3
```

*Figura 86. Ejemplo de uso del operador SDO_FILTER*

### 3.10.2.3 Operador SDO_RELATE

Como se mencionó en el apartado anterior, el operador SDO_FILTER devuelve el resultado exacto más otras geometrías candidatas a cruzarse con la geometría de consulta. El operador SDO_RELATE ejecuta además del primer filtro el segundo, por lo que obtiene el número exacto de resultados.

Solo puede utilizarse si el índice espacial se ha creado en dos dimensiones. Su sintaxis es la siguiente:

```
SDO_RELATE
(
        table_geometry IN SDO_GEOMETRY,
        query_geometry IN SDO_GEOMETRY
        [, parameter_string IN VARCHAR2 ]
) = 'TRUE'
```

*Figura 87. Sintaxis operador SDO_RELATE*

Los parámetros ‘table_geometry’ y ‘query_geometry’ se encuentran descritos en el apartado “*3.10.2.1 Sintaxis genérica de un operador*”. Los parámetros específicos que se pueden indicar en ‘parameter_string’ son:

- querytype=<valor cadena>: establecido siempre como ‘querytype=window’. Es opcional a partir de la versión 10g, sin embargo en versiones anteriores es obligatorio indicarlo.

- mask=<valor cadena>: parámetro obligatorio en el que se indica el tipo de relación topológica que interesa consultar. Existen nueve tipos a establecer TOUCH, OVERLAPBDYDISJOINT, OVERLAPBDYINTERSECT, EQUAL, INSIDE, COVEREDBY, CONTAINS, COVERS, ANYINTERACT y ON. Se explicarán con más detalle antes de ver algunos ejemplos de uso.

- min_resolution=<valor numérico>: Se da el mismo uso que en el operador SDO_FILTER, véase el apartado “*3.10.2.2 Operador SDO_FILTER*”.

- max_resolution=<valor numérico>: Se da el mismo uso que en el operador SDO_FILTER, véase el apartado “*3.10.2.2 Operador SDO_FILTER*”.

Antes de ver algunos ejemplos concretos de uso, se va a hacer un resumen de los tipos de relación topológica disponibles para establecer en el parámetro ‘mask’.

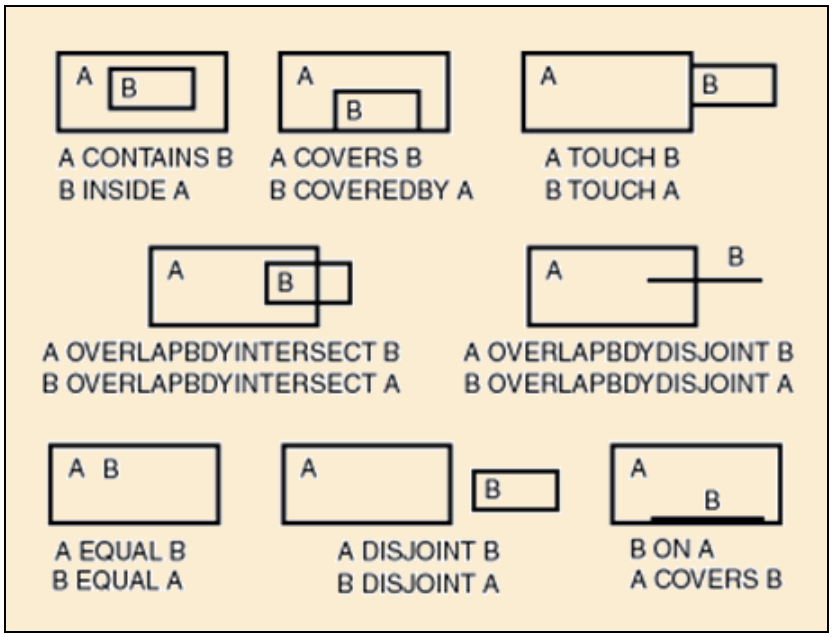



*Figura 88. Tipos de relaciones topológicas [KGE, 2007]*

Donde:

- DISJOINT: ni los límites ni el interior de las geometrías se cruzan.

- TOUCH: los límites se cruzan, pero los interiores no.

- OVERLAPBDYDISJOINT: el interior de un objeto se cruza con el límite e interior del otro objeto, pero los dos límites no se cruzan. Esta relación se da

por ejemplo, cuando una línea se origina fuera de un polígono y finaliza dentro del polígono.

- OVERLAPBDYINTERSECT: los límites e interiores de los dos objetos se cruzan.
- EQUAL: los dos objetos tienen el mismo límite e interior.
- CONTAINS: el interior y el límite de uno de los objetos están completamente contenidos en el interior del otro.
- COVERS: el interior de uno de los objetos está completamente contenido en el interior o el límite del otro objeto y además los límites de los objetos se cruzan.
- INSIDE: lo opuesto a CONTAINS. 'A' dentro de 'B' implica que 'B' contiene a 'A'.
- COVEREDBY: lo opuesto a COVERS. 'A' cubierto por 'B' implica que 'B' cubre a 'A'.
- ON: el interior y el límite de un objeto están en el límite del otro objeto (y el segundo objeto cubre al primero). Esta relación se da por ejemplo cuando una línea está en el límite de un polígono.
- ANYINTERACT: los objetos no son disjuntos.

Una vez entendidos los tipos de relaciones topológicas que pueden darse entre objetos, se van a presentar algunos ejemplos de uso del operador SDO_RELATE, basados en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*".

En primer lugar, se van a recuperar los puntos contenidos en el polígono con código igual a '2'.

```
SELECT pt.cd_punto, pt.descripcion
FROM poligonos pl, puntos pt
WHERE pl.cd_poligono = 2 AND
        MDSYS.SDO_RELATE (pt.geometria, pl.geometria,'mask=INSIDE querytype=WINDOW')
= 'TRUE'

CD_PUNTO            DESCRIPCION
--------------------------  ----------------------
2                   p2
1                   p1
```

*Figura 89. Ejemplo uso operador SDO_RELATE con máscara INSIDE*

A continuación, se consultan los polígonos cuyo límite e interior se cruzan con el interior de la línea con código igual a '1', pero donde los dos límites (el de la línea y el polígono) no se cruzan.

```
SELECT pl.cd_poligono, pl.descripcion
FORM lineas l, poligonos pl
WHERE l.cd_linea = 1 AND
MDSYS.SDO_RELATE(pl.geometria, l.geometria,'mask=OVERLAPBDYDISJOINT
                                querytype=WINDOW' ) = 'TRUE'

CD_POLIGONO          DESCRIPCION
---------------------------  ----------------------
3                    g3
```

*Figura 90. Ejemplo uso operador SDO_RELATE con máscara OVERLAPBDISJOINT*

Por último, se quieren encontrar las líneas que son iguales a la línea con código igual a ‘2’.

```
SELECT l.cd_linea, l.descripcion
FROM lineas lc, lineas l
WHERE lc.cd_linea = 2 AND
        MDSYS.SDO_RELATE(l.geometria, lc.geometria,'mask=EQUAL querytype=WINDOW' )
        = 'TRUE' AND
       l.cd_linea <> 2

CD_LINEA             DESCRIPCION
---------------------------  ----------------------
3                    l3
```

*Figura 91. Ejemplo uso operador SDO_RELATE con máscara EQUAL*

### 3.10.2.4 Operador SDO_WITHIN_DISTANCE

Este operador es el más simple. Sirve para consultar lugares que se encuentran a una determinada distancia de otro lugar. Solo puede utilizarse si el índice espacial se ha creado en dos dimensiones.

El operador SDO_WITHIN_DISTANCE especifica una distancia ‘d’ de la ubicación de la geometría de consulta. El índice espacial va a recuperar las geometrías que estén dentro de esa distancia y va a eliminar los que estén fuera.

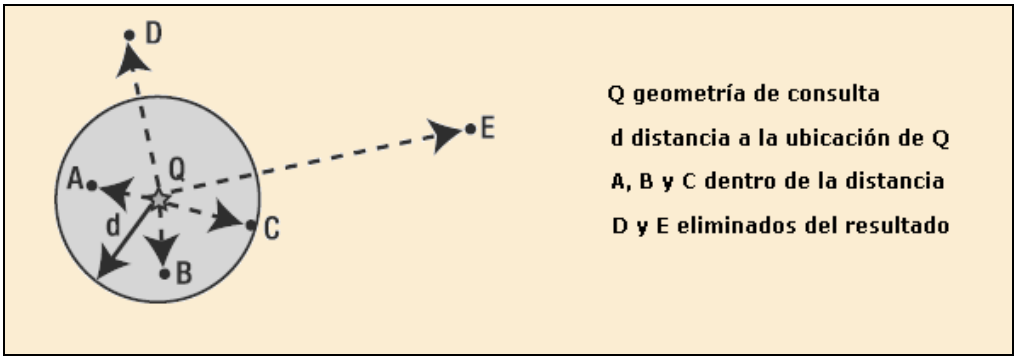



*Figura 92. Distancia “d” en el operador SDO_WITHIN_DISTANCE [KGE, 2007]*

La sintaxis del operador SDO_WITHIN_DISTANCE es la siguiente:

```
SDO_WITHIN_DISTANCE
(
        table_geometry IN SDO_GEOMETRY,
        query_geometry IN SDO_GEOMETRY
        [, parameter_string IN VARCHAR2 ]
) = 'TRUE'
```

*Figura 93. Sintaxis operador SDO_FILTER*

Los parámetros ‘geometry’ y ‘query_geometry’ se encuentran descritos en el apartado “*3.10.2.1 Sintaxis genérica de un operador*”. Los parámetros específicos que se pueden indicar en ‘parameter_string’ son:

- distance=<valor numérico>: parámetro obligatorio en el que se indica la distancia a la geometría de consulta.

- unit=<valor cadena>: parámetro opcional en el que se indica la unidad de medida de la distancia. Los valores disponibles se pueden obtener consultando la tabla MDSYS.SDO_DIST_UNITS.

- min_resolution=<valor numérico>: Se da el mismo uso que en el operador SDO_FILTER, véase el apartado “*3.10.2.2 Operador SDO_FILTER*”.

- max_resolution=<valor numérico>: Se da el mismo uso que en el operador SDO_FILTER, véase el apartado “*3.10.2.2 Operador SDO_FILTER*”.

Como ejemplo, basado en el escenario de la figura “*Figura 79. Escenario de trabajo de operadores y funciones*” se van a recuperar los polígonos que están a una distancia de ‘3’ unidades del punto con código ‘1’.

```
SELECT pl.cd_poligono, pl.descripcion
FROM poligonos pl, puntos pt
WHERE pt.cd_punto = 1 AND
        MDSYS.SDO_WITHIN_DISTANCE (pl.geometria, pt.geometria, 'DISTANCE=3') =
'TRUE'
CD_POLIGONO         DESCRIPCION
--------------------------  ----------------------
2                           g2
1                           g1
3                           g3
```

*Figura 94. Ejemplo de uso del operador SDO_WITHIN_DISTANCE*

### 3.10.2.5 Operador SDO_NN

El operador SDO_WITHIN_DISTANCE recupera todos los objetos que están dentro de una distancia a la geometría de consulta. En ocasiones será más interesante obtener los objetos más cercanos a otro dado. Para dicho propósito SDO_WITHIN_DISTANCE no sirve, ya que el objeto más cercano puede estar a una distancia mayor que la indicada en el operador. SDO_NN recupera los objetos en orden de distancia de la geometría de consulta.

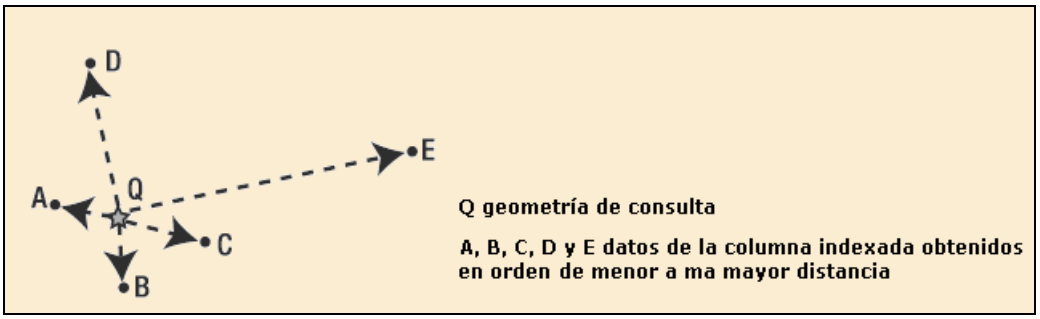



*Figura 95. Orden resultados operador SDO_NN [KGE, 2007]*

Solo puede utilizarse si el índice espacial se ha creado en dos dimensiones. La sintaxis del operador SDO_NN es la siguiente:

```
SDO_NN
(
        table_geometry IN SDO_GEOMETRY,
        query_geometry IN SDO_GEOMETRY
        [, parameter_string IN VARCHAR2
        [, tag IN NUMBER ]]
) = 'TRUE'
```

*Figura 96. Sintaxis operador SDO_NN*

Los parámetros ‘geometry’ y ‘query_geometry’ se encuentran descritos en el apartado “*3.10.2.1 Sintaxis genérica de un operador*”. Los parámetros específicos que se pueden indicar en ‘parameter_string’ son:

- distance=<valor numérico>: parámetro opcional utilizado para restringir la distancia a partir de la cual se detiene la búsqueda, es decir buscará los objetos más cercanos pero dentro de una distancia.

- unit=<valor cadena>: parámetro opcional en el que se indica la unidad de medida de la distancia. Los valores disponibles se pueden obtener consultando la tabla MDSYS.SDO_DIST_UNITS.

- sdo_batch_size=<valor numérico>: cuando en la cláusula WHERE de la consulta se combina el uso del operador SDO_NN con otros predicados no espaciales, para encontrar no solo los objetos más cercanos, sino los más cercanos que cumplan además una condición, se puede dar el caso de que el operador SDO_NN tenga que evaluarse varias veces hasta que los resultados satisfagan las condiciones indicadas en la cláusula WHERE. Internamente el índice aplica el operador sobre un bloque de filas que establece por defecto, si el número de resultados obtenido al evaluar las filas del bloque no satisface la cláusula WHERE, el índice vuelve a aplicarse al siguiente bloque, así sucesivamente hasta que el número de resultados obtenidos satisfaga la cláusula WHERE. Si es conocido que los resultados buscados están dentro de un determinado número de objetos más cercanos, se puede establecer el tamaño del bloque para hacer que la consulta sea más rápida y eficiente.

- sdo_num_res=<valor numérico>: normalmente no interesa recuperar toda la lista de objetos en orden de distancia a la geometría de consulta, sino un

número determinado de objetos más cercanos, por ejemplo los cinco lugares más cercanos a otro dado. Como se verá en los ejemplos, esto se puede conseguir haciendo uso del predicado 'ROWNUM<=N', pero estableciendo un valor en este parámetro el operador SDO_NN devuelve exactamente los 'N' objetos más cercanos y puede ser evaluado con mayor rapidez que sin el parámetro. Existe también otra diferencia a tener en cuenta, 'sdo_mnum_res=N' devuelve los 'N' objetos más cercanos con un orden en el resultado que no tiene porque corresponder con el orden de la distancia a la geometría de consulta. Mientras que haciendo uso del predicado 'ROWNUM<=N' los resultados si se obtienen ordenados de menos a mayor distancia a la geometría de consulta. Es importante tener en cuenta que si se ha establecido el parámetro 'sdo_batch_size' el parámetro 'sdo_mnum_res' es ignorado.

En la sintaxis del operador, se observa que acompañando a los parámetros 'sdo_batch_size' y 'sdo_mnum_res' se puede indicar el parámetro 'tag'. Cuando se buscan los objetos más cercanos a otro dado puede ser interesante saber la distancia a la que se encuentran cada uno de ellos. Dado que el operador SDO_NN internamente calcula las distancias para identificar los objetos más cercanos, esto se puede conseguir sin coste adicional haciendo uso de esta etiqueta (tag) y el operador auxiliar SDO_NN_DISTANCE. Este operador auxiliar se especifica como parte de la lista SELECT y está destinado a un operador SDO_NN en la cláusula WHERE. Se verá con más claridad mediante los ejemplos.

Basado en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*" se trata de recuperar los tres polígonos más cercanos al punto con código '3'. En primer lugar se va a realizar haciendo uso del parámetro 'sdo_num_res'. Seguidamente haciendo uso del predicado ROWNUM.

```
SELECT pl.cd_poligono, pl.descripcion
FROM poligonos pl, puntos pt
WHERE pt.cd_punto = 3 AND
        MDSYS.SDO_NN(pl.geometria, pt.geometria, 'SDO_NUM_RES=3') = 'TRUE'

CD_POLIGONO          DESCRIPCION
-------------------------    ----------------------
2                    g2
3                    g3
4                    g4
```

*Figura 97. Ejemplo uso parámetro SDO_NUM_RES del operador SDO_NN*

```
SELECT pl.cd_poligono, pl.descripcion
FROM poligonos pl, puntos pt
WHERE pt.cd_punto = 3 AND
        MDSYS.SDO_NN(pl.geometria, pt.geometria, 'SDO_BATCH_SIZE=10') = 'TRUE'
      …AND ROWNUM<=3
CD_POLIGONO          DESCRIPCION
-------------------------    ----------------------
3                    g3
4                    g4
2                    g2
```

*Figura 98. Ejemplo uso predicado ROWNUM<=N en el operador SDO_NN*

Comparando el resultado obtenido en las figuras "*Figura 97. Ejemplo uso parámetro SDO_NUM_RES del operador SDO_NN*" y "*Figura 98. Ejemplo uso predicado ROWNUM<=N en el operador SDO_NN*" se puede observar que el orden en los resultados no es el mismo.

Haciendo uso del cuarto parámetro 'tag' y el operador auxiliar SDO_NN_DISTANCE, se va a comprobar que el predicado 'ROWNUM<=N' muestra los resultados en orden de distancia, mientras que el parámetro SDO_NUM_RES no lo hace.

```
SELECT pl.cd_poligono, pl.descripcion, MDSYS.SDO_NN_DISTANCE(1) distancia
FROM poligonos pl, puntos pt
WHERE pt.cd_punto = 3 AND
MDSYS.SDO_NN(pl.geometria, pt.geometria, 'SDO_NUM_RES=3',1) = 'TRUE'

CD_POLIGONO         DESCRIPCION          DISTANCIA
--------------------------- ---------------------- ----------------------
2                   g2                   3
3                   g3                   1.5
4                   g4                   2
```

*Figura 99. Ejemplo uso operador auxiliar SDO_NN_DISTANCE*

Observando la sentencia anterior se ve que en SDO_NN_DISTANCE se indica entre paréntesis la etiqueta numérica '1'. Esta etiqueta numérica se establece como cuarto parámetro en el operador SDO_NN y sirve para enlazar el operador auxiliar SDO_NN_DISTANCE a la instancia del operador SDO_NN de la cláusula WHERE. Como resultado junto con los datos de cada polígono se obtiene la distancia de la instancia a la geometría de consulta (el punto con código '3').

## 3.10.3 Funciones

En el apartado anterior se han visto los índices espaciales y los operadores como herramienta de análisis de la información espacial. Spatial proporciona también las funciones espaciales. Éstas se diferencian de los operadores en lo siguiente:

- No hacen uso de los índices espaciales, por lo que no es necesario tener un índice creado para utilizarlas. A su vez esto implica que las consultas sean más lentas y costosas.

- Ofrecen un análisis más detallado de la información espacial que los operadores.

- Pueden ser usados tanto en la cláusula WHERE como en la lista de selección SELECT.

En los siguientes apartados se presentan algunas de las más utilizadas.

### 3.10.3.1 FUNCIONES BUFFER

La primera función a tratar es SDO_BUFFER. Ésta función genera un buffer alrededor de un objeto geométrico especificado o a un conjunto de objetos geométricos.

La función SDO_BUFFER, puede trabajar tanto con objetos simples como con objetos SDO_GEOMETRY complejos, cómo un compuesto de polígonos o colecciones.

En el caso de objetos geométricos con un agujero interior, el buffer generado también tendrá el mismo agujero interior.

La sintaxis es la siguiente:

```
SDO_BUFFER
(
        geometry        IN SDO_GEOMETRY,
        distance        IN NUMBER,
        tolerance       IN NUMBER
        [, params       IN VARCHAR2]
)
RETURNS an SDO_GEOMETRY
```

*Figura 100. Sintaxis SDO_BUFFER*

Dónde:

- geometry: determina el objeto SDO_GEOMETRY sobre el que vamos a generar el buffer.

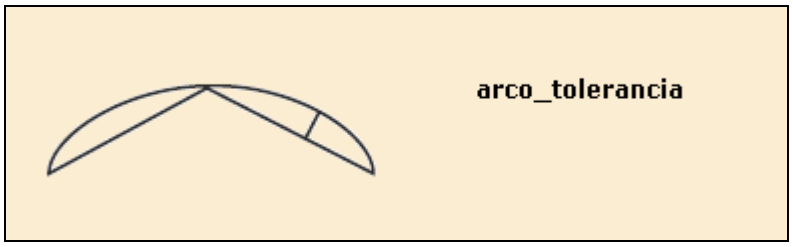
- distance: determina la distancia numérica al buffer de la geometría entrante. Si el valor es positivo, el búfer se genera en torno a la geometría, si el valor es negativo (válido sólo para los polígonos), el búfer se genera dentro de la geometría

- *tolerance*: determina la tolerancia a utilizar.

- params: son los parámetros opcionales de la función. Al igual que para los operadores, se indican expresados en modo nombre_parametro=valor_parametro:

    - unit=<valor cadena>: indica la unidad de medida de la distancia. Los valores disponibles se pueden obtener consultando la tabla MDSYS.SDO_DIST_UNITS.

    - arc_tolerance=<valor numérico>: es obligatorio si la geometría es geodésica, es decir, cuando el SRID de la geometría se establece a un SRID geodésico como 8307 u 8265. En un espacio geodésico los arcos no están permitidos, son representados usando aproximaciones lineales. Este parámetro determina la distancia máxima entre el arco y su aproximación lineal.




Antes de ver algún ejemplo, con respecto al parámetro opcional arc_tolerance es importante tener en cuenta lo siguiente:

- El valor indicado en arc_tolerance siempre debe ser mayor que la tolerancia de la geometría.

- La tolerancia de los geodésicos se especifica en metros. El parámetro arc_tolerance, sin embargo, siempre se especifica en las unidades que determine el parámetro "unit".

Basado en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*" se va a generar un buffer sobre el polígono con código igual a '3' a una distancia de tres unidades.

Para ello se va a crear una nueva tabla en la que introducir la geometría buffer. Dado que se trabaja en el mismo escenario se asignan los mismos metadatos que a la tabla POLIGONOS.

```
CREATE TABLE PRUEBA_BUFFER AS
SELECT CD_POLIGONO AS CD_BUFFER,
SDO_GEOM.SDO_BUFFER(p.geometria, 3, 0.25) AS GEOMETRIA
FROM poligonos p
WHERE p.cd_poligono = 3

INSERT INTO user_sdo_geom_metadata
SELECT 'PRUEBA_BUFFER', 'GEOMETRIA', diminfo, srid
FROM mdsys.user_sdo_geom_metadata
WHERE table_name = 'POLIGONOS'
```

*Figura 101. Ejemplo de uso de la función SDO_BUFFER*

Como se puede observar en la figura anterior, no se ha indicado el parámetro opcional que indica la unidad de medida de la distancia. Si se omite el parámetro la unidad que utilizada es la unidad de medida asociada con los datos. En éste escenario de trabajo, centímetros, que es la unidad correspondiente al sistema de referencia con SRID '262144'.

Consultando la tabla PRUEBA_BUFFER, se puede observar la geometría correspondiente al buffer generado.

```
SELECT * FROM PRUEBA_BUFFER;

CD_BUFFER   GEOMETRIA
----------  ----------------------------------------------------------------------------------------------
3           SDO_GEOMETRY
            (       2003, 262144, NULL,
                    SDO_ELEM_INFO_ARRAY
                    (       1, 1005, 8, 1, 2, 2, 5, 2, 1, 7, 2, 2, 11, 2, 1,
                             13, 2, 2, 17, 2, 1, 19, 2, 2, 23, 2, 1
                    ),
                    SDO_ORDINATE_ARRAY
                    (       1.5, 4.5, 2.37867966, 2.37867966, 4.5, 1.5, 8.5, 1.5,
                            10.6213203, 2.37867966,11.5, 4.5, 11.5, 6, 10.6213203,
                            8.12132034, 8.5, 9, 4.5, 9, 2.37867966, 8.12132034, 1.5,
                            6, 1.5, 4.5
                    )
            )
```

*Figura 102. Ejemplo de geometría buffer obtenida mediante la función SDO_BUFFER*

## 3.10.3.2 FUNCIONES DE ANÁLISIS DE RELACIÓN

Las funciones de análisis de relación sirven para consultar la distancia entre geometrías o el tipo de relación topológica que hay entre ellas.

A continuación se presentan las de uso más común.

### 3.10.3.2.1 SDO_DISTANCE

La función SDO_DISTANCE, obtiene la mínima distancia entre dos puntos de dos geometrías.

La sintaxis del operador es la siguiente:

```
SDO_DISTANCE
(
        geometry1       IN SDO_GEOMETRY,
        geometry2       IN SDO_GEOMETRY,
        tolerance       IN NUMBER,
        [, params       IN VARCHAR2]
)
RETURN      a NUMBER
```

*Figura 103. Sintaxis SDO_DISTANCE*

Dónde:

- geometry1 y geometry2: objetos SDO_GEOMETRY entre los que se quiere calcular la distancia.
- tolerance: determina la tolerancia a utilizar en el cálculo. Es importante establecerla adecuadamente para evitar errores de redondeo.

- params: son los parámetros opcionales de la función. Al igual que para los operadores, se indican expresados en modo nombre_parametro=valor_parametro;

    - unidad=<valor cadena>: determina el tipo de unidad de medida en el que se devuelve la distancia. Se pueden obtener los valores posibles para las unidades de la tabla MDSYS.SDO_DIST_UNITS.

Esta función devuelve la mínima distancia entre 'geometry1' y 'geometry2', en las unidades especificadas. Si no se especifica ninguna, por defecto se utilizará el del sistema de coordenadas de los datos.

Se recomienda el uso de la función SDO_DISTANCE sólo:

- Para operar con tablas no indexadas
- Para complementar al operador SDO_WITHIN_DISTANCE

Como ejemplo, se van a recuperar los polígonos que se encuentran a una distancia máxima de 20 milímetros del punto con código '3'.

```
SELECT pl.cd_poligono, pl.descripcion
FROM puntos pt, poligonos pl
WHERE pt.cd_punto = 3 AND
        SDO_GEOM.SDO_DISTANCE(pl.geometria, pt.geometria, 0.25, 'unit=MM') <= 20
ORDER BY pl.cd_poligono

CD_POLIGONO          DESCRIPCION
-----------------------  -------------------------
3                    g3
4                    g4
```

*Figura 104. Ejemplo de uso de la función SDO_DISTANCE*

Basado en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*", en la que la unidad de medida son centímetros, se puede observar que efectivamente los polígonos que se encuentran a una distancia máxima de 20 milímetros (2 centímetros) son las que se muestran en el resultado de la consulta. Si no se hubiese indicado la unidad de medida, dado que por defecto usa la de los datos, en este caso centímetros, se hubiese tenido que establecer la condición a '<=2'.

#### 3.10.3.2.2 RELATE

Sirve para consultar relaciones topológicas entre geometrías. La sintaxis de la función es la siguiente:

```
RELATE
(
        geometry_A          IN SDO_GEOMETRY,
        mask                IN VARCHAR2,
        geometry_Q          IN SDO_GEOMETRY,
        tolerance           IN NUMBER
)
RETURNS relación de tipo VARCHAR2
```

*Figura 105. Sintaxis función RELATE*

Dónde:

- geometry_A, geometry_Q: son los objetos SDO_GEOMETRY entre los que se va a analizar la relación.
- mask: tipos de máscara a establecer que puede tener uno de los siguientes valores:
    - DETERMINE: Determina la relación o interacción que geometry_A tiene con geometry_Q
    - Tipo de relación topológica que quiere consultar. Se pueden utilizar las mismas que las indicadas en el operador SDO_RELATE, véase la figura "*Figura 88. Tipos de relaciones topológicas*".
    - ANYINTERACT: Si se mantiene algún tipo de relación topológica diferente a DISJOINT.

La función RELATE devuelve:

- 'TRUE', si hay intersección entre las geometrías y ha indicado en el parámetro 'mask' en tipo de relación topológica ANYINTERACT.
- Si 'geometry_A' cumple el tipo de relación topológica especificado en la relación con 'geometría_Q', la función devuelve la relación topológica.
- 'FALSE' si la relación entre las geometrías no coincide con la relación especificada en el parámetro 'mask'.
- El tipo de relación topológica, si el parámetro 'mask' se fija a DETERMINE

Se debe usar la función RELATE, sólo en algunos escenarios, por ejemplo, para operar con tablas no indexadas o también para complementar al operador SDO_RELATE.

A continuación se presentan algunos ejemplos basados en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*".

En primer lugar, se trata de recuperar los polígonos que tienen algún tipo de relación con el polígono con código '2'.

```
SELECT pl2.cd_poligono, pl2.descripcion
FROM poligonos pl2, poligonos pl
WHERE pl.cd_poligono = 2 AND
        SDO_GEOM.RELATE(pl2.geometria, 'ANYINTERACT', pl.geometria, 0.25) = 'TRUE'
ORDER BY pl2.cd_poligono

CD_POLIGONO          DESCRIPCION
-----------------------  ----------------------
1                        g1
2                        g2
3                        g3
```

*Figura 106. Ejemplo de uso función RELATE con relación ANYINTERACT*

Observando la figura del escenario de trabajo se comprueba que ‘g1’ está dentro de ‘g2’ (INSIDE), ‘g2’ es igual a sí mismo (EQUAL) y el límite e interior de ‘g3’ se cruza con el límite e interior de ‘g2’ (OVERLAPBDYINTERSECT). El resto de polígonos no se cruzan con el límite o interior de ‘g2’, es decir, son disjuntos, por lo que no se obtienen como resultados de la consulta.

En este segundo ejemplo se va a indicar una relación topológica determinada para encontrar las líneas contenidas en el polígono con código ‘4’.

```
SELECT l.cd_linea, l.descripcion
FROM poligonos p, lineas l
WHERE p.cd_poligono = 4 AND
         SDO_GEOM.RELATE(p.geometria, 'CONTAINS', l.geometria, 0.25) = 'CONTAINS'
ORDER BY l.cd_linea

CD_LINEA       DESCRIPCION
-----------------  ----------------------
2                  l2
3                  l3
```

*Figura 107. Ejemplo de uso función RELATE con relación CONTAINS*

Observando la figura del escenario se observa que ‘l2’ y ‘l3’ están dentro de ‘g4’, es decir que ‘g4’ las contiene.

Por último se va a hacer uso del valor fijo DETERMINE, para recuperar la relación topológica existente entre la línea con código ‘1’ y el resto de polígonos del escenario.

```
SELECT p.cd_poligono, p.descripcion,
        SDO_GEOM.RELATE(p.geometria, 'DETERMINE', l.geometria, 0.25) as relacion
FROM poligonos p, lineas l
WHERE l.cd_linea = 1
ORDER BY p.cd_poligono

CD_POLIGONO          DESCRIPCION          RELACION
----------------------  ----------------------  ------------------------------------
1                       g1                      DISJOINT
2                       g2                      TOUCH
3                       g3                      OVERLAPBDYDISJOINT
4                       g4                      DISJOINT
5                       g5                      DISJOINT
```

*Figura 108. Ejemplo de uso función RELATE con máscara DETERMINE*

### 3.10.3.3 FUNCIONES DE COMBINACIÓN DE GEOMETRÍAS

En matemáticas, dos conjuntos de elementos 'A' y 'B', pueden ser combinados usando diferentes teorías de conjuntos, Spatial proporciona funciones similares que actúan sobre un par de geometrías en lugar de sobre un par de conjuntos. Son las siguientes:

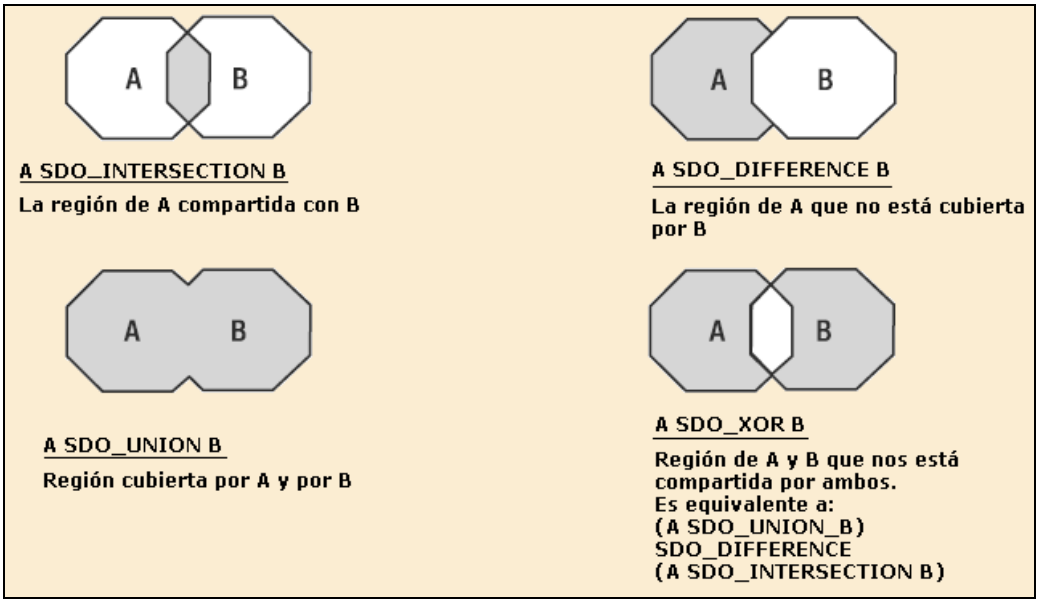



*Figura 109. Funciones de combinación de geometrías [KGE, 2007]*

La sintaxis de una función de combinación es la siguiente:

```
SDO_<set_theory_fn>
(
        geometry_A      IN SDO_GEOMETRY,
        geometry_B      IN SDO_GEOMETRY,
        tolerance       IN NUMBER
)
RETURNS SDO_GEOMETRY
```

*Figura 110. Sintaxis función combinación*

Donde:

- geometry_A, geometry_B: son los objetos SDO_GEOMETRY a combinar. Ambos deben presentar el mismo sistema de referencia (SRID).

- tolerance: valor de la tolerancia a aplicar para los objetos SDO_GEOMETRY.

Antes de ver algún ejemplo de uso, es importante señalar que solo son aplicables en sistemas en dos dimensiones, por ejemplo, no se podría realizar la unión de dos sólidos.

Basado en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*", se van a ver algunos ejemplos de combinación.

En primer lugar se va a calcular la intersección del polígono con código '3' y la línea con código '1'.

```
SELECT SDO_GEOM.SDO_INTERSECTION(p.geometria, l.geometria, 0.25) AS RES_INTERS
FROM poligonos p, lineas l
WHERE p.cd_poligono = 3 AND l.cd_linea = 1

RES_INTERS
--------------------------------------------------------------------------------------------------------------------------------
SDO_GEOMETRY
        (       2002, 262144, NULL,
                SDO_ELEM_INFO_ARRAY(1, 2, 1),
                SDO_ORDINATE_ARRAY(6, 4,5, 6, 6)
        )
```

*Figura 111. Ejemplo de uso de la función SDO_INTERSECTION*

Observando el escenario se puede comprobar que se obtiene una línea que corresponde al tramo de la línea 'l' que tiene intersección con el polígono 'g3'.

En este segundo ejemplo se va a calcular la unión entre los polígonos con códigos '4' y '5'.

```
SELECT SDO_GEOM.SDO_UNION(p.geometria, p2.geometria, 0.25) AS RES_UNION
FROM poligonos p, poligonos p2
WHERE p.cd_poligono = 4 AND p2.cd_poligono = 5

RES_UNION
--------------------------------------------------------------------------------------------------------------------------------
SDO_GEOMETRY
        (       2007, 262144, NULL,
                SDO_ELEM_INFO_ARRAY
                        (1, 1003, 1, 15, 1003, 1),
                SDO_ORDINATE_ARRAY
                (       12.5, 4, 13.5, 3, 15, 3, 16, 4, 15, 5, 13.5, 5, 12.5, 4,
                        10, 6, 10, ,5, 12.5, ,5, 12.5, 4, 10, 6
                )
        )
```

*Figura 112. Ejemplo de uso de la función SDO_UNION*

Los polígonos 'g4' y g5' se tocan en un punto pero no se cruzan, por lo que el resultado que se obtiene corresponde a un multipolígono formado por los dos polígonos.

Seguidamente en este ejemplo se busca la diferencia entre los polígonos con códigos '2' y '1'.

```
SELECT SDO_GEOM.SDO_DIFFERENCE(p.geometria, p2.geometria, 0.25) AS RES_DIFF
FROM poligonos p, poligonos p2
WHERE p.cd_poligono = 2 AND p2.cd_poligono = 1

RES_DIFF
--------------------------------------------------------------------------------------------------------------------------------
---
SDO_GEOMETRY
        (       2003, 262144, NULL,
                SDO_ELEM_INFO_ARRAY
                        (1, 1003, 1, 11, 2003, 1),
                SDO_ORDINATE_ARRAY
                        (       1, 7, 1, 1, 5, 1, 5, 7, 1, 7,
                                2, 3, 2, 4.5, 3.5, 4.5, 4, 3, 2, 3
                        )
        )
```

*Figura 113. Ejemplo de uso de la función SDO_DIFFERENCE*

Como se puede observar en la figura del escenario, ‘g1’ está dentro de ‘g2’, por lo que como resultado se obtiene el polígono con agujero correspondiente a ‘g2’ con ‘g1’ como agujero.

Por último, se va a ver un ejemplo de la función SDO_XOR sobre los polígonos con códigos ‘2’ y ‘3’.

```
SELECT SDO_GEOM.SDO_XOR(p.geometria, p2.geometria, 0.25) AS RES_XOR
FROM poligonos p, poligonos p2
WHERE p.cd_poligono = 2 AND p2.cd_poligono = 3;

RES_ XOR
--------------------------------------------------------------------------------------------------------------------------------
SDO_GEOMETRY
        (       2007, 262144, NULL,
                SDO_ELEM_INFO_ARRAY
                        (1, 1003, 1, 11, 1003, 1),
                SDO_ORDINATE_ARRAY
                        (       5, 6, 5, 4.5, 8.5, 4.5, 8.5, 6, 5, 6,
                                1, 7, 1, 1, 5, 1, 5, 4.5, 4.5, 4.5, 4.5, 6, 5, 6, 5, 7, 1, 7
                        )
)
```

*Figura 114. Ejemplo de uso de la función SDO_XOR*

Como resultado se obtiene el multipolígono formado por el polígono ‘g2’ menos la región que comparte con ‘g3’ y el polígono ‘g3’ menos la región que comparte con ‘g2’.

## 3.10.3.4 FUNCIONES DE ANÁLISIS GEOMÉTRICO

En el apartado anterior se ha visto como construir geometrías que representan la intersección, unión o diferencia de geometrías. Spatial proporciona otra serie de funciones de análisis geométrico a aplicar sobre geometrías individuales o sobre el resultado de otras funciones, por ejemplo uniones o intersecciones.

La sintaxis de este tipo de funciones es la siguiente:

```
function_name
(
        geometry                IN SDO_GEOMETRY,
        tolerance               IN NUMBER
        [, units_params                 IN VARCHAR2]
)
RETURN NUMBER
```

*Figura 115. Sintaxis función análisis geométrico*

Donde:

- geometry: especifica el objeto SDO_GEOMETRY a ser analizado.
- tolerance: valor de la tolerancia a aplicar durante el cálculo.
- units_params: parámetro opcional que especifica la unidad de medida en que se devuelve el área o longitud. Se indica de forma nombre_parametro=valor_parametro. Se pueden obtener unidades de medida de distancia y área de las tablas MDSYS.SDO_DIST_UNITS y MDSYS.SDO_AREA_UNITS respectivamente. No hay tabla definida para unidades de volumen, lo que significa que no se realizan operaciones de conversión en el cálculo de volúmenes.

A continuación se van a presentar algunas de las funciones por separado y un ejemplo de cada una de ellas basado en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*".

### 3.10.3.4.1 SDO_AREA

Sirve para calcular el área del objeto SDO_GEOMETRY. Tiene sentido aplicarlo para polígonos, superficies o sólidos (o colecciones). Para puntos o líneas el área es cero.

Como ejemplo, se va a calcular el área de la intersección de los polígonos con códigos '2' y '3'.

```
SELECT SDO_GEOM.SDO_AREA(
                                    SDO_GEOM.SDO_INTERSECTION(
                                            p.geometria, p2.geometria, 0.25
                                    ), 0.25
                            ) AS AREA
FROM poligonos p, poligonos p2
WHERE p.cd_poligono = 2 and p2.cd_poligono = 3;

AREA
----------
0.75
```

*Figura 116. Ejemplo de uso de la función SDO_AREA*

Se puede comprobar que la intersección de los polígonos ‘g2’ y ‘g3’ es un rectángulo de la lados 1.5 y 0.5. El área es 1.5 * 0.5 = 0.75, que es el resultado obtenido en la consulta. Al no establecer unidad de medida se obtiene en la correspondiente a la de los datos, es decir centímetros cuadrados.

### 3.10.3.4.2 SDO_LENGTH

Calcula la longitud de una línea o perímetro de un polígono, superficie o sólido. Para puntos devuelve siempre cero.

Se va a ver un ejemplo calculando la longitud de la línea con código ‘2’ en milímetros.

```
SELECT SDO_GEOM.SDO_LENGTH(l.geometria, 0.25, 'unit=MM') AS LONGITUD
FROM lineas l
WHERE l.cd_linea= 2


LONGITUD
----------
15
```

*Figura 117. Ejemplo de uso de la función SDO_LENGTH*

Observando el escenario se ve que la línea ‘l2’ tiene una longitud de 1.5 centímetros, al establecer como unidad de medida el milímetro “MM” la consulta devuelve el resultado en milímetros.

### 3.10.3.4.3 SDO_VOLUME

Calcula el volumen de un sólido o multisólido. Para el resto de geometrías la función devuelve cero.

Como ejemplo se va a calcular el volumen de un sólido consistente en un cubo optimizado.

```
SELECT SDO_GEOM.SDO_VOLUME
(           SDO_GEOMETRY
            (           3009,NULL,NULL,
                        SDO_ELEM_INFO_ARRAY
                        (           1,1007,3,
                                    7,1007,3
                        ),
                        SDO_ORDINATE_ARRAY
                        (           -2.0, 1.0, 3.0, -3.0, -1.0, 0.0,
                                     0.0, 0.0, 0.0, 1.0, 1.0, 1.0
                        )
            ), 0.005) AS VOLUMEN
FROM DUAL;

VOLUMEN
-------------------
6
```

*Figura 118. Ejemplo de uso de la función SDO_VOLUME*

# 3.11 Visualización de datos espaciales

En este último apartado, se va a tratar un aspecto muy importante, la visualización de los objetos de tipo SDO_GEOMETRY usando mapas.

Los puntos a tratar en este apartado son los siguientes:

- En primer lugar se va a presentar la herramienta de visualización de mapas proporcionada por Oracle, Map Viewer.
- Seguidamente, se analizarán los elementos que componen los mapas y la relación entre los mismos.
- Tras haber especificado los elementos de construcción de mapas, se van a indicar los mecanismos para definirlos en la base de datos (Map Definition Tool), basado en el escenario utilizado anteriormente para describir los operadores. Ver figura "*Figura 79. Escenario de trabajo de operadores y funciones*".
- Por último se nombrarán las técnicas principales usadas desde las aplicaciones para interactuar con mapas. Y se realizará una petición XML simple que muestre el escenario definido en el apartado anterior.

## 3.11.1 Definición de la herramienta MapViewer

MapViewer es un componente de Java, que construye mapas a partir de la información de las tablas que contienen los datos espaciales, y determinadas vistas que definen el formato apropiado en el que deben mostrarse.

En la figura se muestran los principales componentes de la misma:

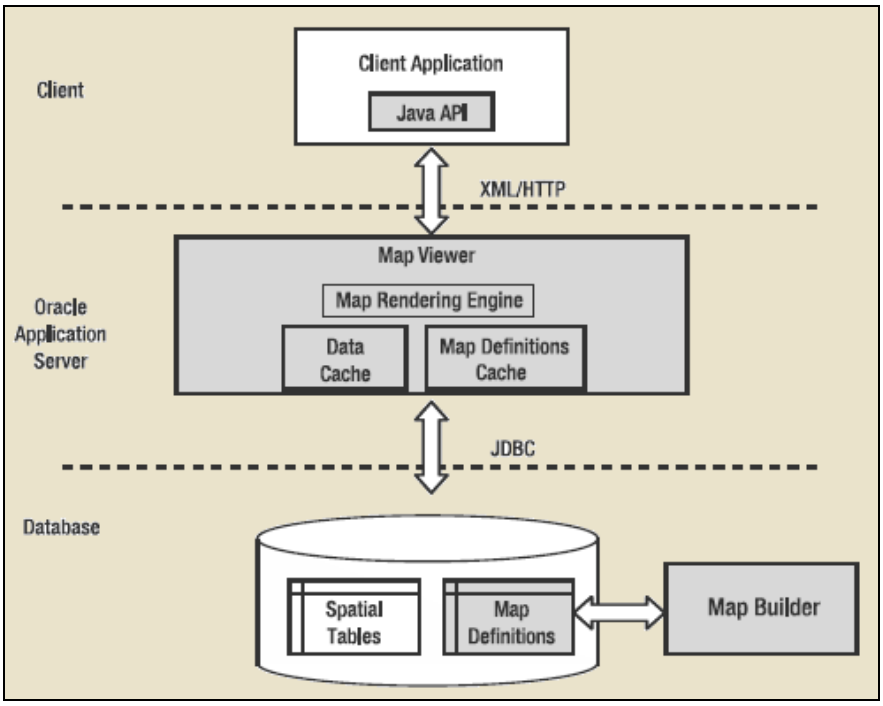



*Figura 119. Componentes MapViewer [KGE, 2007]*

- El servlet Java que procesa las peticiones enviadas por aplicaciones cliente, obtiene de la base de datos la información pertinente y construye los mapas que son devueltos al cliente con la posibilidad de distintos formatos.

- Definición de los mapas. Son almacenados en la base de datos y sirven para describir las características de los mapas, que tablas usar, que estilos, colores, etc.

- Interfaces de programación de aplicaciones: las APISs permiten acceder a MapViewer desde una variedad de entornos de desarrollo de aplicaciones, tales como XML, Java, PL/SQL y AJAX. Simplifican la construcción y parseo de peticiones y respuestas al servidor. El API de Java incluye librerías de etiquetas que facilitan la inclusión de mapas en los JSPs.

- Herramienta de definición de mapas: es un programa que ayuda al usuario a manejar las definiciones de los mapas almacenados en la base de datos.

Las aplicaciones cliente se comunican con el servlet de MapViewer sobre un modelo de petición y respuesta HTTP. Las peticiones y las respuestas son codificadas en XML, aunque también es posible devolver directamente la imagen generada a partir de la petición.

Puesto que MapViewer es una herramienta puramente de Java, puede utilizarse en cualquier plataforma donde Java esté disponible. Sin embargo esto no implica que solo pueda utilizarse con aplicaciones Java, sino con cualquiera que soporte XML sobre HTTP.

Se puede descargar de la página de la OTN de Oracle y disponer de ella simplemente instalando el contenedor de servlets para J2EE OC4J. Se necesita la plataforma Java J2SE SDK 1.5 o posterior.

Una vez instalado el fichero de despliegue 'mapviewer.ear' es muy fácil verificar si se ha desplegado correctamente. Basta con escribir en el navegador la siguiente URL http://127.0.0.1:8888/mapviewer. Debería mostrarse la página de inicio de MapViewer:

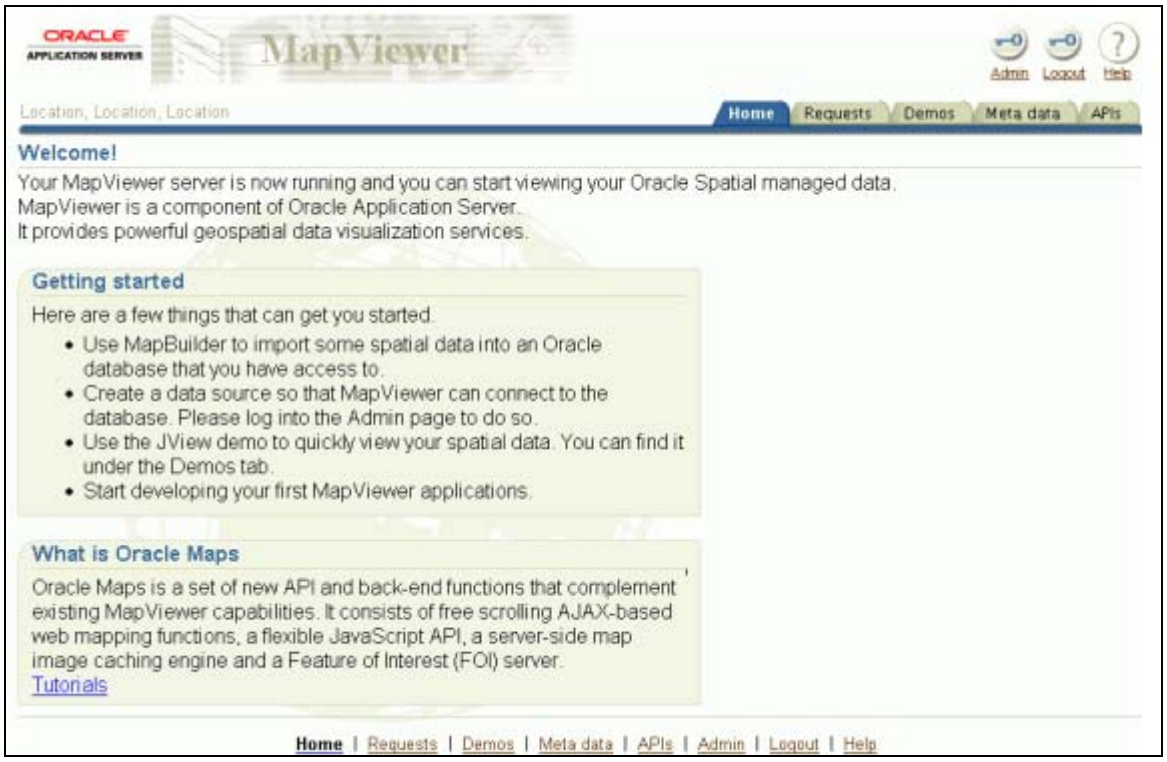



*Figura 120. Página de Bienvenida de MapViewer*

Desde está página se podrá administrar la configuración de MapViewer, parámetros, fuentes de datos, etc., ver demos, crear sus peticiones XML, manejar APIs de Javascript de interacción con mapas, etc. Oracle proporciona la guía de referencia User's Guide for Oracle MapViewer 11g, donde se puede consultar toda la información detallada de MapViewer.

## 3.11.2 Elementos de construcción de mapas

Hasta el momento se ha dado una visión de en qué consiste MapViewer y como realizar la instalación del mismo, pero ¿cómo se definen los mapas que se generan a partir de las peticiones realizadas a MapViewer?

En la figura se muestran los elementos a manejar a la hora de definir los mapas.

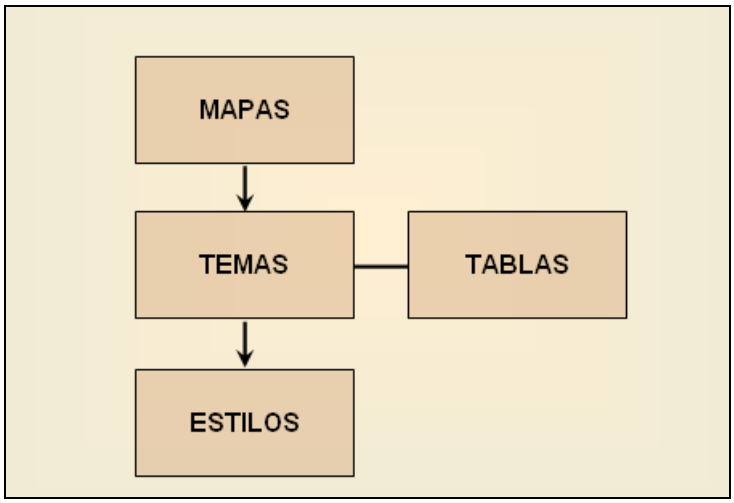



*Figura 121. Elementos de construcción de mapas*
*(Elaboración propia)*

Los temas definen que información aparece en el mapa, mientras que los estilos describen cómo se representan. La figura anterior es una simplificación. Un mapa se compone de uno o varios temas, un mismo tema es utilizado por unos o varios mapas y varios temas pueden basarse en una misma tabla; a su vez un tema utiliza varios estilos y un mismo estilo puede ser usado por varios temas.

Antes de entrar en detalle en la construcción de mapas, es conveniente ver con más detalle los conceptos de estilo y tema en los siguientes apartados.

### 3.11.2.1 ESTILOS

Para dibujar la información espacial almacenada en la base de datos, es necesario indicarle a MapViewer cómo hacerlo, es decir que estilo usar para representarla. Esto se consigue asociando a cada tema, un estilo.

Un estilo es un atributo visual que representa una característica espacial. La elección de estos atributos es muy importante a la hora de dar mayor legibilidad al mapa, por ello es muy común usar símbolos y atributos definidos por estándares a los que los usuarios están habituados, como por ejemplo los usados en los mapas de carreteras, tales como el dibujo de un surtidor de gasolina para indicar un punto en el que se encuentra una gasolinera, los distintos tipos de líneas para identificar los tipos de carretera, etc.

Hay seis tipos de estilos:

- COLOR: color para el borde (STROKE) o relleno (FILL).
- MARKER: figura geométrica (con un color de relleno y borde) o imagen para representar puntos.
- LINE: usado para representar características lineales, definiendo tamaño, color, centro y borde de las líneas. También se puede definir como terminan las líneas y como se representan las intersecciones entre las mismas.

- AREA: patrones de relleno para áreas.
- TEXT: tipo de fuente, color y efectos de brillo para etiquetas
- ADVANCED: composición de estilos.

Son almacenados en base de datos en la vista USER_SDO_STYLE, que puede ser modifica por cada usuario para definir y mantener sus estilos personalizados. En la vista ALL_SDO_STYLES se pueden consultar todos los estilos definidos por los usuarios de la base de datos. No se aplican privilegios a esta vista, una vez definido en la vista USER_SDO_STYLE, puede ser utilizado por cualquier usuario de la base de datos.

La estructura de la vista USER_SDO_STYLE es la siguiente:

| Campo | Descripción |
|---|---|
| NAME | Nombre del estilo |
| TYPE | Tipo del estilo (Uno de los seis disponibles). |
| DESCRIPCION | Descripción del estilo. |
| DEFINITION | Fichero XML de definición del estilo. |
| IMAGE | Fichero de imagen para cuando el tipo de estilo es MARKER |

*Tabla 9. Estructura de a vista USER_SDO_STYLE*

### 3.11.2.2 TEMAS

Tal y como se ha indicado en el apartado anterior los estilos se aplican a los distintos temas que conforman un mapa. A continuación se va a explicar en qué consisten dichos temas.

Los mapas se construyen mediante temas. En sistemas SIG o herramientas de generación de mapas reciben el nombre de capas. Se puede pensar en una capa como una celda en la que se van a dibujar un conjunto de geometrías según una serie de criterios. El mapa se obtiene situando cada capa una encima de otra siguiendo un orden. Este es el mismo método usado en cartografía tradicional durante siglos.

Un tema es un concepto muy potente que puede usarse para agrupar objetos espaciales en subconjuntos lógicos. Ejemplos típicos de temas que se encuentran en las aplicaciones son:

- Fronteras: fronteras de ciudades, países, estados, provincias…
- Características naturales: ríos, lagos, bosques…
- Rutas de transportes: caminos, calles, vías de tren…
- Localización de clientes o almacenes…

El modo de agrupar los objetos espaciales en temas es muy importante, puesto que determina la manera en que se va a interactuar con el mapa. Los usuarios pueden decidir qué datos quieren ver en un mapa, pero seleccionando temas completos. Por lo que los objetos a mezclar dentro de un mismo tema tienen que guardar una relación lógica útil.

Esto no significa que tengan que ser objetos homogéneos, un ejemplo típico se da en la hidrografía para representar lagos y ríos en un mismo mapa mediante polígonos y líneas respectivamente.

También se pueden definir múltiples temas sobre una misma tabla, realizando la consulta adecuada sobre la misma para cada uno de ellos. Por ejemplo, en el caso de que se haya definido una tabla con todas las carreteras, independientemente del tipo de las mismas (autopista, autovía, carretera secundaria…), sería interesante crear un tema para cada una de ellas según su tipo, para representarlas de forma diferente dentro del mapa.

Tras definir que es un tema, cabe preguntarse donde se almacena los temas en la base de datos de Oracle.

La vista USER_SDO_THEMES se puede modificar para que cada usuario pueda definir sus temas personalizados. En la vista ALL_SDO_THEMES se pueden consultar todos los temas definidos por los usuarios en tablas, sobre las que el usuario que realiza la consulta, tenga permisos.

La estructura de la vista es la siguiente:

| Campo | Descripción |
|---|---|
| NAME | Nombre del tema |
| DESCRIPCION | Descripción del tema. |
| BASE_TABLE | Nombre de la tabla de la que se obtienen los objetos espaciales que componen el tema. |
| GEOMETRY_COLUMN | Nombre de la columna de la tabla correspondiente al objeto espacial. |
| STYLING_RULES | Definición XML del tema. |

*Tabla 10. Vista USER_SDO_THEMES*

## 3.11.3 Definición de mapas

Tal y como se ha definido en el apartado anterior, un mapa es una colección de temas. Sin embargo construir un mapa, es más que listar los temas que deben aparecer en el mapa. El orden en el que se añaden los temas es importante. Además, la definición del mapa posibilita controlar el conjunto de información a incluir, dependiendo de la escala del mismo. Estos dos conceptos orden y escala son muy importantes y se van a tratar a continuación.

### 3.11.3.1 ORDEN

Los temas de un mapa deben ser ensamblados en el orden correcto dependiendo del tipo de simbología de cada uno y su importancia relativa. Supongase que en el escenario de ejemplo de operadores y funciones, se incluye primero la capa para los puntos y después la capa de los polígonos. La capa de los polígonos podría difuminar u ocultar algunos de los componentes de la capa puntos.

Map Viewer renderiza los temas en el mismo orden en el que se introducen en la lista, es decir el primer tema de la lista es renderizado primero y así sucesivamente de modo que el tema del final de la lista es renderizado el último.

Nótese que se pueden usar temas transparente en determinadas momentos, para que otros temas se visualicen correctamente sin necesidad de colocarlos en el principio de la lista.

### 3.11.3.2 ESCALA Y NIVEL DE ZOOM

La escala determina el área geográfica a mostrar. Representa la relación entre una distancia en el mapa y la distancia real de la misma sobre el terreno. Por ejemplo si 2cm en el mapa representan 1Km sobre el suelo, la escala sería 2cm/1Km, que es lo mismo que 2cm/100.000cm o 1/50.000, es decir 1:50.000.

Cuanto más grande es la escala, menos área se presenta con más nivel de detalle; y cuanto más pequeña es la escala más grande es el área que se presenta y menor el nivel de detalle.

Hacer zoom sobre un mapa, no es más que cambiar la escala, es decir se pregunta por el mismo mapa, producido a diferente escala.

La cantidad de información a mostrar en un mapa depende de la escala del mismo. Supóngase que en un mapa se muestra España entera, y un usuario hace zoom sobre una provincia en concreto para verla con más detalle, a medida que vaya haciendo zoom, irán desapareciendo algunas capas, como por ejemplo el resto de provincias e irán apareciendo otras como por ejemplo los pueblos de la provincia que se quiere observar más de cerca. Esto se consigue estableciendo para cada tema un rango de escala. Un tema solo aparecerá en el mapa, cuando la escala en la que esté el mismo esté dentro del rango del tema. Por tanto es muy importante establecer correctamente la escala.

### 3.11.3.3 Vista USER_SDO_MAPS

Al igual que todos los elementos definidos anteriormente, los mapas son definidos por los usuarios en una vista propia de Oracle USER_SDO_MAPS. En la vista ALL_SDO_MAPS se pueden consultar todos los mapas definidos por los usuarios en tablas, sobre las que el usuario que realiza la consulta, tenga permisos.

La estructura de la vista USER_SDO_MAPS es la siguiente:

| Campo | Descripción |
|---|---|
| NAME | Nombre del mapa |
| DESCRIPCION | Descripción del mapa |
| DEFINITION | Definición XML del mapa. |

*Tabla 11. Vista USER_SDO_MAPS*

## 3.11.4 Manejo de los elementos de construcción de mapas

Tal y como se ha mencionado anteriormente, los usuarios definen los elementos que componen los mapas en las vistas SDO_USER_STYLES, SDO_USER_THEMES, SDO_USER_MAPS. La actualización de los datos de las mismas, se puede realizar usando sentencias SQL estándar o procesos de importación de datos almacenados en otros esquemas o ficheros.

Pero además Oracle proporciona herramientas de definición de mapas con una interfaz sencilla, que permiten manejar los datos de forma cómoda, tales como ‘Map Builder’ o ‘Map Definition Tool’.

Estas herramientas están disponibles en la OTN de Oracle. Son librerías .jar, que solo requiere los drivers JDBC de Oracle, para establecer la conexión a nuestro esquema de base de datos de trabajo.

A continuación y basado en el escenario definido en el ejemplo de funciones y operadores, ver figura “*Figura 79. Escenario de trabajo de operadores y funciones*”, se va a dar una visión general del manejo de una de las herramientas “Map Definition Tool”.

En la documentación de referencia de Oracle, se pueden encontrar guías completas y detallas de ambas herramientas.

Una vez descargado el fichero ‘mapdef.jar’, para entrar en la herramienta, solo es necesario ejecutar el siguiente comando:

```
java -cp $ORACLE_HOME\lib\classes12.jar;mapdef.jar oracle.eLocation.console.GeneralManager
```

*Figura 122. Ejecución Oracle Definition Tool*

Es importante recordar que se necesitan los drivers JDBC de Oracle “classes12.jar”. Una vez ejecutado el comando se mostrará la pantalla de conexión.

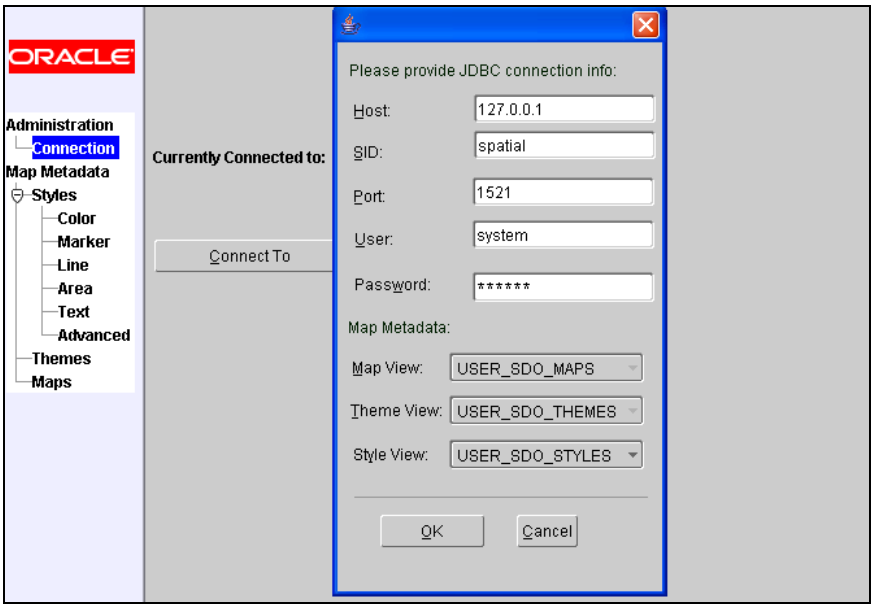



*Figura 123. Pantalla acceso Map DefinitionTool*

Tras realizar la conexión con la fuente de datos indicada, se muestra la pantalla desde la administrar los estilos, temas y mapas.

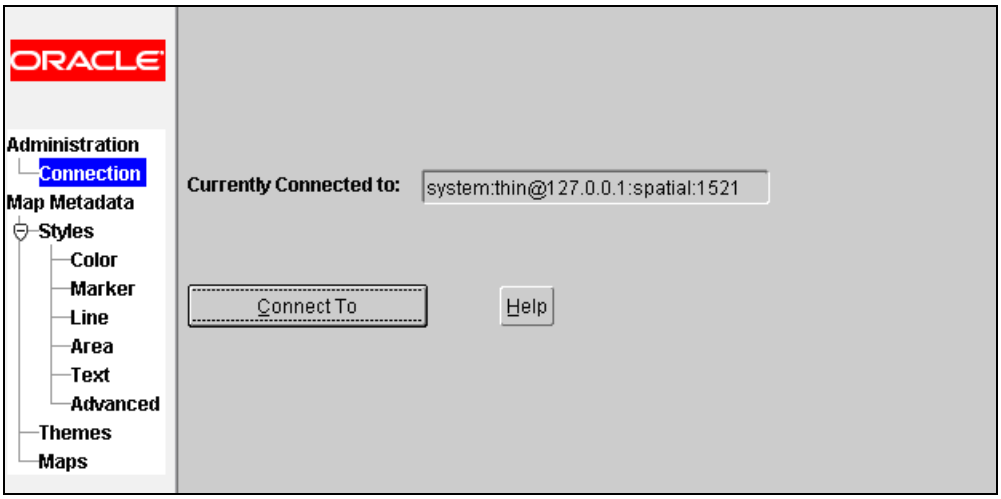



*Figura 124. Pantalla datos conexión Map DenitionTool*

Todas las pantallas tienen una estructura similar y la misma forma de manejo sencilla. En la parte izquierda de la ventana aparece el menú de los elementos a administrar. Tras seleccionar el tipo de elemento deseado, en la parte derecha se mostrarán los elementos disponibles y un panel de edición de datos de un elemento concreto.

Se va a describir el funcionamiento de la pantalla correspondiente a la administración de estilos, pero el manejo es el mismo en el caso de los temas y mapas.

Oracle proporciona una gran variedad de estilos predeterminados para los diferentes tipos, al seleccionar el tipo de estilos que desea administrar dichos estilos se cargan en la lista de elementos disponibles:

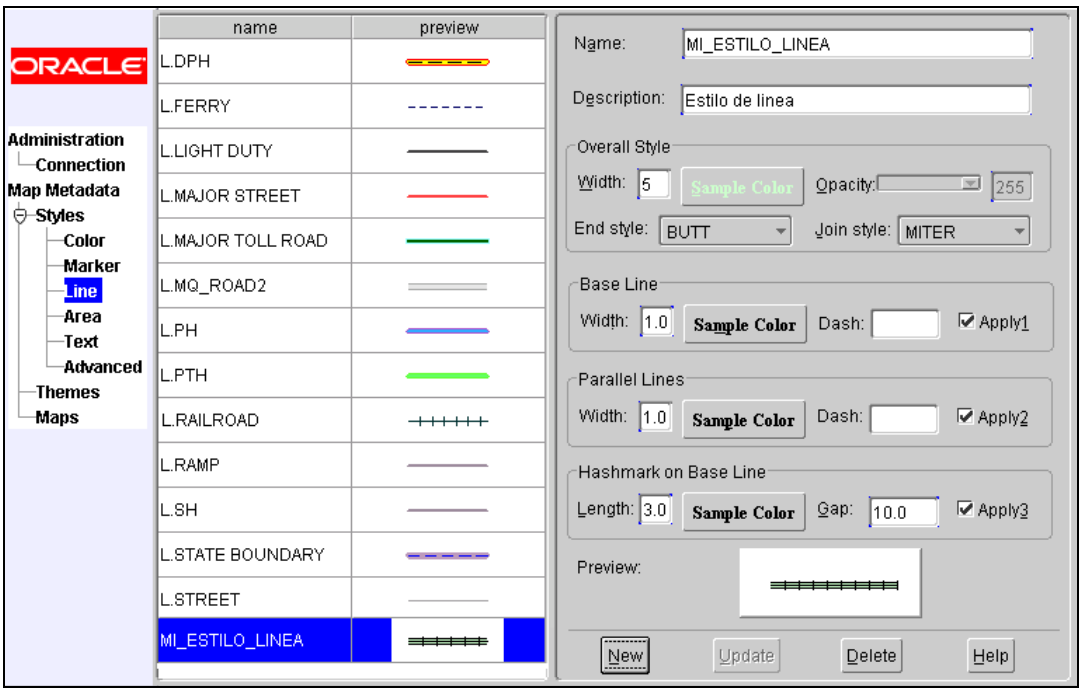



*Figura 125. Pantalla definición de estilos de Map Definition Tool*

Como se había mencionado anteriormente, al seleccionar el elemento a administrar en el menú izquierdo, en este caso estilos de tipo línea, aparecen los ya existentes. El manejo es muy sencillo:

- Para crear un nuevo elemento pulse el botón ‘New’, se rellenan los datos en el panel de edición del elemento y se pulsa el botón “Insert”. El nuevo elemento insertado aparece en la lista de elementos disponibles.

- Para modificar un elemento se selecciona en la lista de disponibles, los datos se cargarán en el panel de edición del elemento. Tras modificar los datos deseados se pulsa el botón ‘Update’.

- Por último para eliminar un elemento, se selecciona en lista de disponibles y se pulsa el botón ‘Delete’.

Una vez vistos los estilos disponibles que se pueden utilizar, se van a definir los temas del mapa de la figura *“Figura 79. Escenario de trabajo de operadores y funciones”*, siguiendo un criterio de división por tipo de geometría. Así pues se van a definir tres temas:

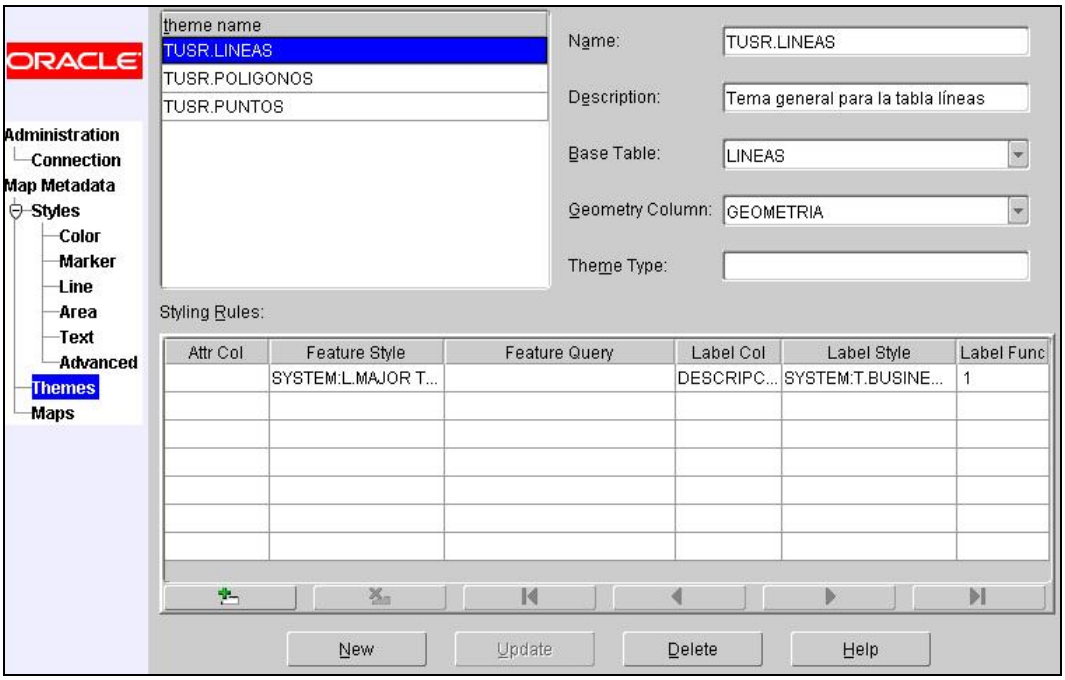



*Figura 126. Pantalla definición de temas de Map Definition Tool*

La definición XML de la regla de estilo a aplicar al tema se presenta en forma de tabla. Se pueden añadir nuevas reglas, modificarlas y borrarlas, editando los datos desde la propia fila de la tabla.

Por cada regla de estilo se muestra:

- Attr Col: Utilizada solo al aplicar un estilo avanzado. Indica el nombre del campo de la tabla base o una expresión SQL sobre el mismo, que controla como se renderiza cada rango establecido en el estilo avanzado. Por ejemplo, en mapas que muestran la densidad de población de un determino país, se puede establecer un estilo avanzado para mostrar cada región de un color según el rango de población de la zona. En esta columna habría que establecer el nombre del campo que contiene la media de población por región.

- Feature Style: Nombre del estilo a usar para la regla de estilo.
- Feature Query: Condición SQL que determina las filas de la tabla base sobre las que aplicar la regla de estilo definida en la fila.
- Label: Además del estilo a aplicar se puede definir una etiqueta. Si es el caso debemos indicar:
    - Label Col: nombre del campo o expresión SQL sobre varios campos de la tabla base, del que se obtiene el texto a mostrar en la etiqueta.
    - Label Style: Nombre del estilo a usar para las etiquetas.
    - Label Func: Valor numérico, expresión SQL o función (que devuelva un valor numérico). Si el valor es mayor que cero, la geometría a la que se aplica la regla de estilo definida se muestra etiquetada, en caso contrario no se etiqueta. Para conseguir que todas las geometrías a las que aplica la regla de estilo se muestras etiquetadas, se puede establecer directamente el valor 1.

En el tema 'TUSR_LINEAS' se van a presentar todas las líneas igual, utilizando el estilo de línea predeterminado L.MAJOR TOLL ROAD' y etiquetando cada línea con el campo 'descripción' de la tabla, usando el estilo de texto predefinido 'T.BUSINESS NAME'.

Los temas 'TUSR_PUNTOS' y 'TUSR_POLIGONOS' se han definido de la misma forma, escogiendo uno de los estilos predefinidos para los tipos 'Marker' y 'Area' respectivamente.

Por último, una vez establecidos los temas, sólo queda definir el mapa correspondiente al escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*".

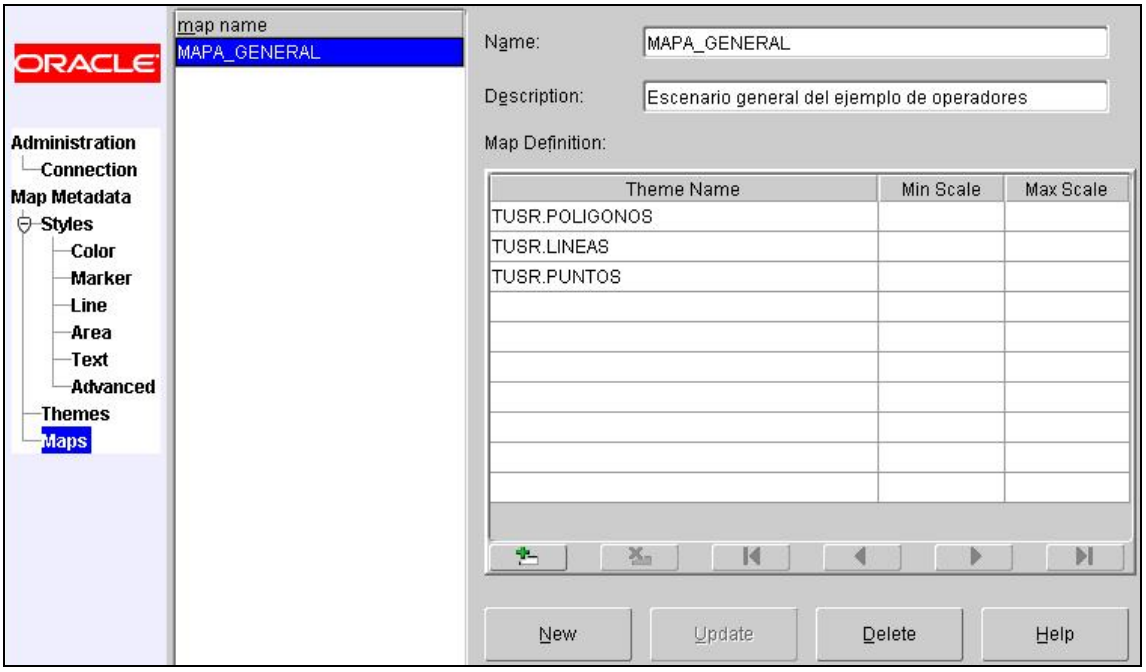



*Figura 127. Pantalla de definición de mapas de Map Definition Tool*

Bajo el nombre y descripción del mapa, se visualiza una tabla en la que se muestran los temas que componen el mapa. El orden establecido en la lista, es el orden el que se renderizan los temas en el mapa, es decir el último tema de la lista se renderiza el último. Se pueden añadir nuevas temas, modificaros o borrarlos editando los datos desde la propia fila de la tabla. Se puede establecer el orden haciendo uso de los botones que se encuentran bajo el listado.

Es importante recordar además que por cada tema puede establecerse un rango de escala. El tema solo aparecerá en el mapa, cuando la escala en la que esté el mismo esté dentro del rango del tema.

El mapa del ejemplo se compone de los temas 'TUSR_POLIGONOS', 'TUSR_LINEAS' y 'TUSR_PUNTOS'. Para evitar que los polígonos oculten las líneas y puntos, se renderizan primero los polígonos, las líneas y por último las capas. Debido al tamaño del escenario no se ha establecido rango de escala para ningún tema.

En el siguiente apartado se verá el resultado de definición del mapa, haciendo uso de una petición XML en MapViewer.

## 3.11.5 Generación de mapas desde aplicaciones

Básicamente, existen tres técnicas disponibles para interactuar con mapas generados por el servlet de MapViewer desde aplicaciones cliente:

- XML API:Usado desde cualquier entorno.
- Java API: Recomendado para programas Java (servlets, applets, jsps…)
- JSP Tags: Páginas JSPs.

El flujo de operación para realizar la petición usando cualquiera de las tres técnicas es la siguiente:

- La aplicación cliente construye una petición a un servicio Web para obtener el mapa. Dicha petición contiene la fuente de base de datos de Oracle de la que se debe leer la información para generar el mapa, el nombre del mapa a generar (debe coincidir con el nombre que se le ha dado al mapa en la vista USER_SDO_MAPS), el formato de salida de la respuesta, el tamaño en píxeles y el área de cobertura del mapa. La petición puede construirse generando manualmente el fichero XML u obtenerlo mediante determinadas llamadas a métodos del API de Java.
- La aplicación cliente llama al servlet de Map Viewer sobre HTTP, pasando la petición XML como parámetro, y de nuevo puede realizarse manualmente o vía API de Java.

- El servlet de MapViewer construye la respuesta XML, que incluye la URL de acceso a la imagen del mapa generada en el servidor, y la devuelve a la aplicación cliente.

En la guía de referencia User's Guide for Oracle MapViewer 11g se puede consultar toda la información detallada de cómo trabajar con la técnica que se adapta a un escenario de trabajo concreto.

Como ejemplo se va a ver el resultado de la definición del mapa para el escenario de trabajo de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*", haciendo uso del API XML.

Desde el formulario de peticiones XML disponible en MapViewer (opción 'Request' de la pantalla de bienvenida de la MapViewer, véase la figura "*Figura 120. Página de Bienvenida de MapViewer*") se introduce la siguiente petición XML:

```
<?xml version="1.0" standalone="yes"?>
<map_request
        title="Ejemplo operadores"
        basemap="MAPA_GENERAL"
        datasource="mvdemo"
        width="640"
        height="480"
        bgcolor="#a6cae0"
        antialiase="false"
        format="PNG_STREAM">
        <center size="16">
                <geoFeature>
                        <geometricProperty>
                                <Point>
                                        <coordinates>8, 4</coordinates>
                                </Point>
                        </geometricProperty>
                </geoFeature>
        </center>
</map_request>
```

*Figura 128. Petición XML MapViewer*

El nodo 'map_request' describe la petición a MapViewer. Los atributos y nodos del mismo son los que definen la forma y aspecto del mapa:

- tittle: texto opcional que aparece como título del mapa si se indica. Por defecto se sitúa encima del mapa, pero puede cambiarse modificando la configuración global de los mapas en el fichero de configuración "mapViewerConfig.xml" de MapViewer.

- basemap: nombre del mapa a desplegar, corresponde al mapa definido en la vista USER_SDO_MAPS.

- datasource: nombre del datasource JDBC del que se obtienen los datos. Debe haberlo configurado previamente.

- width y height: tamaño en píxeles de la imagen resultante.

- format: formato de la imagen a generar. MapViewer puede generar los mapas en los formatos gráficos GIF, PNG o JPEG. Además puede devolver la imagen de una de las siguientes formas:

    - Como una URL en el XML de respuesta, especificando en el atributo "format" GIF_URL, PNG_URL, o JPEG_URL.

    - Directamente el stream de la imagen, especificando en el atributo "format" GIF_STREAM, PNG_STREAM, o JPEG_STREAM.

- bgcolor: color de fondo de la imagen de, mapa. Por defecto es 'ocean blue'.

- antialiase: si se establece a true, MapViewer obtiene el mapa con mejor calidad gráfica, a cambio de tardar más en generarlo.

Todavía es necesario especificar el área que se incluirá en el mapa. Para ello, hay que especificar el centro del mapa y su tamaño mediante el nodo . El atributo 'size' indica el tamaño y el contenido del nodo 'coordinates' las coordenadas 'x' e 'y' correspondientes al centro del mapa.

Dado que en el ejemplo de petición se ha utilizado el formato PNG_STREAM, al realizar el submit de la petición se obtendrá directamente la imagen del mapa.

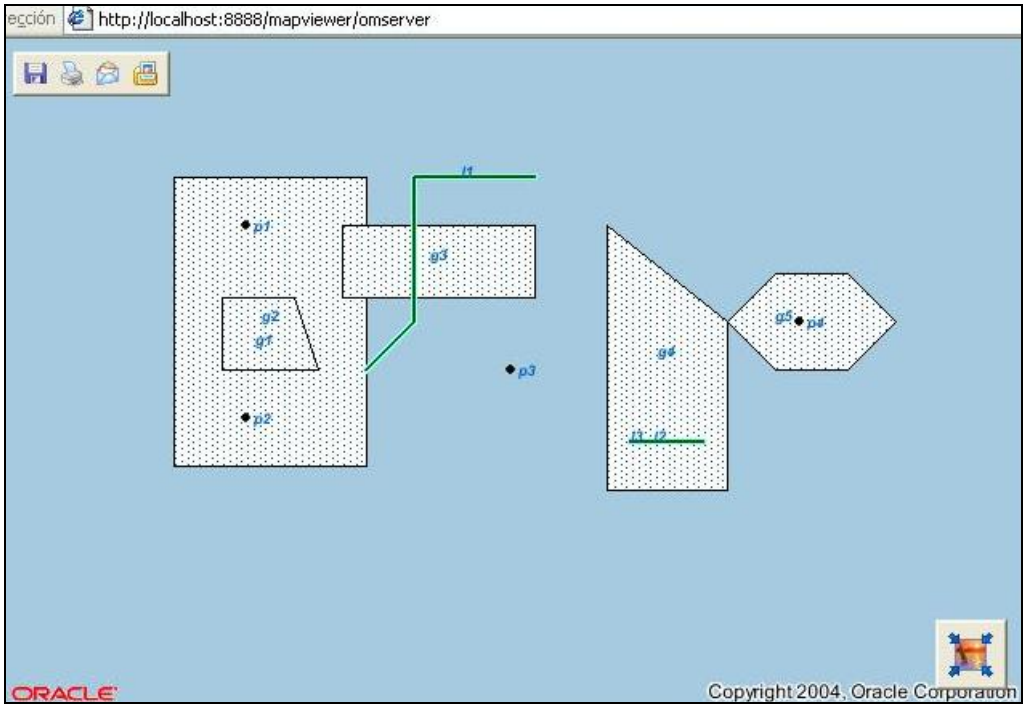



*Figura 129. Ejemplo submit petición XML en MapViewer*

Se puede comprobar que las geometrías corresponden a las dibujadas en el escenario de la figura "*Figura 79. Escenario de trabajo de operadores y funciones*" y que cada tema se renderiza con el estilo especificado en el apartado anterior, "*3.11.4 Manejo de los elementos de construcción de mapas*".

# Capítulo 4

# Aplicación

En este apartado se incluye la información correspondiente al análisis, diseño, implementación y evaluación del aplicativo ADGSO (Aplicación Didáctica de Generación de Geometrías en Spatial de Oracle).

La documentación incluida, se ha generado en base al estándar de la *IEEE 1058.1 – 1987* [IEEE, 1987] para el desarrollo de planes de proyectos software.

El aplicativo ADGSO es un software pequeño que no requiere un reparto de tareas distribuidas entre un equipo de trabajo coordinado, por lo que se ha decidido hacer uso del modelo en cascada realimentado, dada la simplicidad y eficacia demostrada en desarrollos de porte pequeño y mediano.

A lo largo de los siguientes apartados se recoge la documentación generada en las distintas fases:

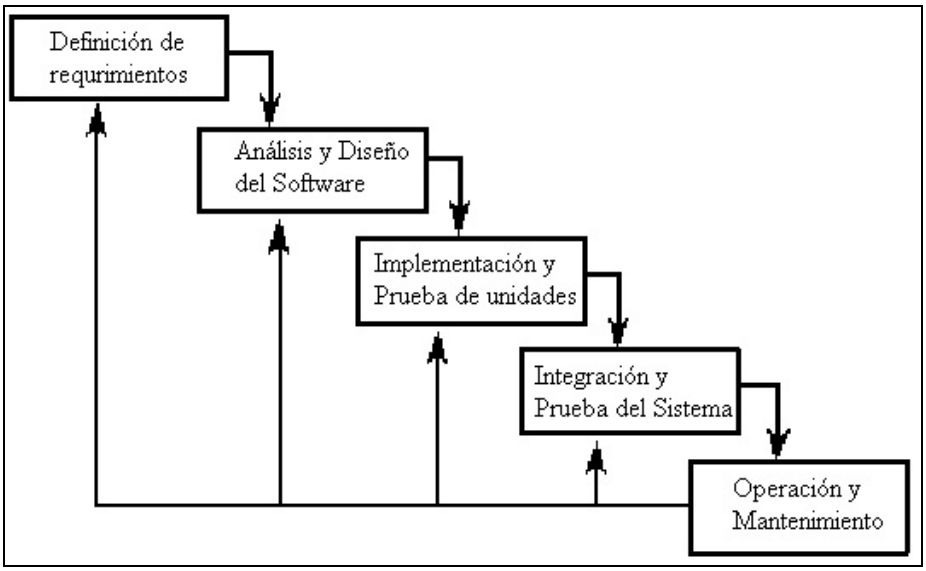



*Figura 130. Esquema del modelo en cascada realimentado*

# 4.1 Especificación de requisitos

Este apartado describe la especificación de requisitos software que debe satisfacer el aplicativo ADGSO.

## 4.1.1 Funciones de la aplicación

El aplicativo a desarrollar tiene una función principalmente didáctica. Consiste en aportar una herramienta que ayude a entender la forma en la que Oracle Spatial guarda la información espacial en la base de datos. A grandes rasgos, la funcionalidad está dividida en tres grupos:

- Creación de capas: ofrece una interfaz para las capas en las que se va a introducir la información espacial.

- Creación de geometrías: ofrece interfaces para definir geometrías indicando el tipo de la misma, los puntos que la componen y el tipo de unión entre los puntos. Tras crear la geometría, muestra la sentencia SQL necesaria para introducir la geometría en la base de datos junto con una tabla explicativa de la misma.

- Validación de geometrías: el usuario puede comprobar si una sentencia de construcción de un determinado tipo de geometría, es correcta, tanto sintáctica como semánticamente.

En el apartado "*4.1.3 Requisitos funcionales específicos*" se verá con más detalle la funcionalidad incluida en cada grupo.

## 4.1.2 Características de los usuarios

El aplicativo a desarrollar está enfocado a dos roles de usuarios, que accederán a la aplicación previa autenticación y tendrán acceso sólo a aquellas funcionalidades que les competan:

- Profesor: el usuario profesor tiene acceso a todas las funcionalidades de la aplicación, tanto a la creación de capas, como a la construcción y validación de geometrías.

- Alumno: el usuario alumno solo tiene acceso a las funcionalidades correspondientes a la manipulación de geometrías, creación y validación.

## 4.1.3 Requisitos funcionales específicos

### 4.1.3.1 Diagrama de casos de uso del rol Profesor

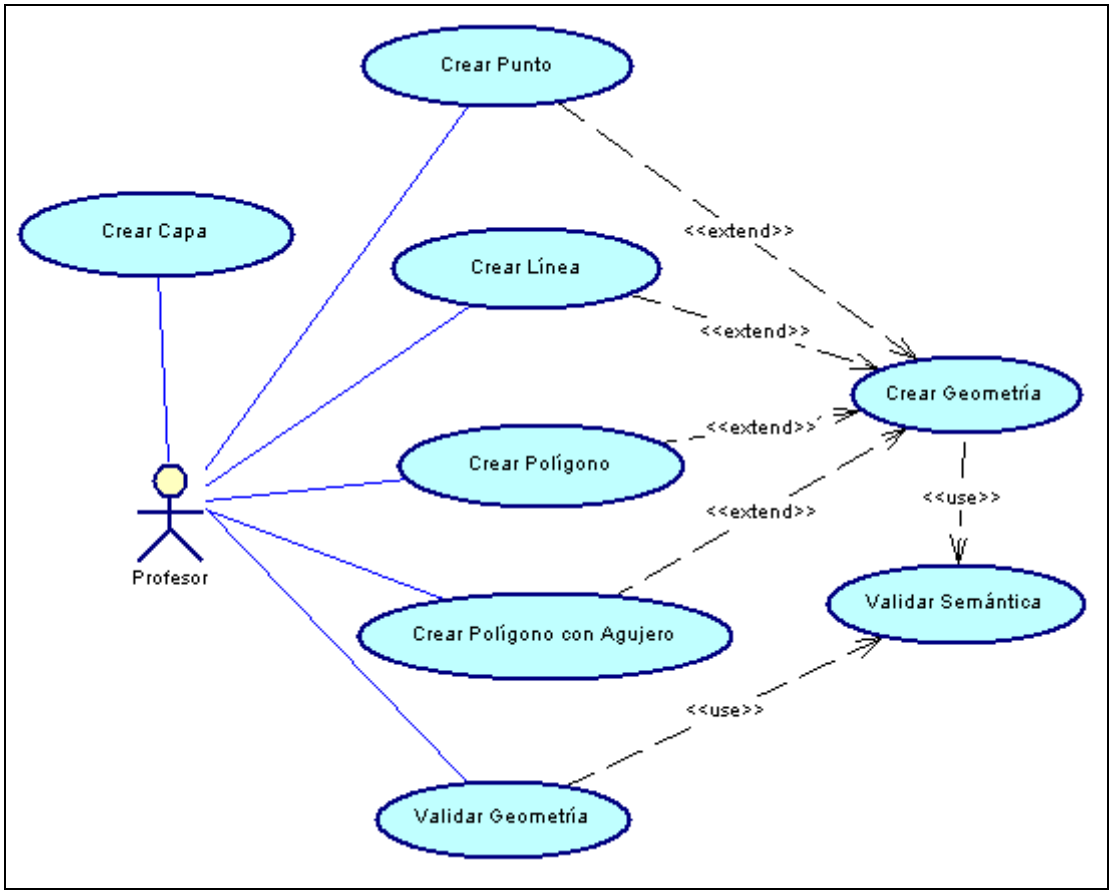



*Figura 131. Diagrama casos de uso del rol profesor*

### 4.1.3.2 Diagrama de casos de uso del rol Alumno

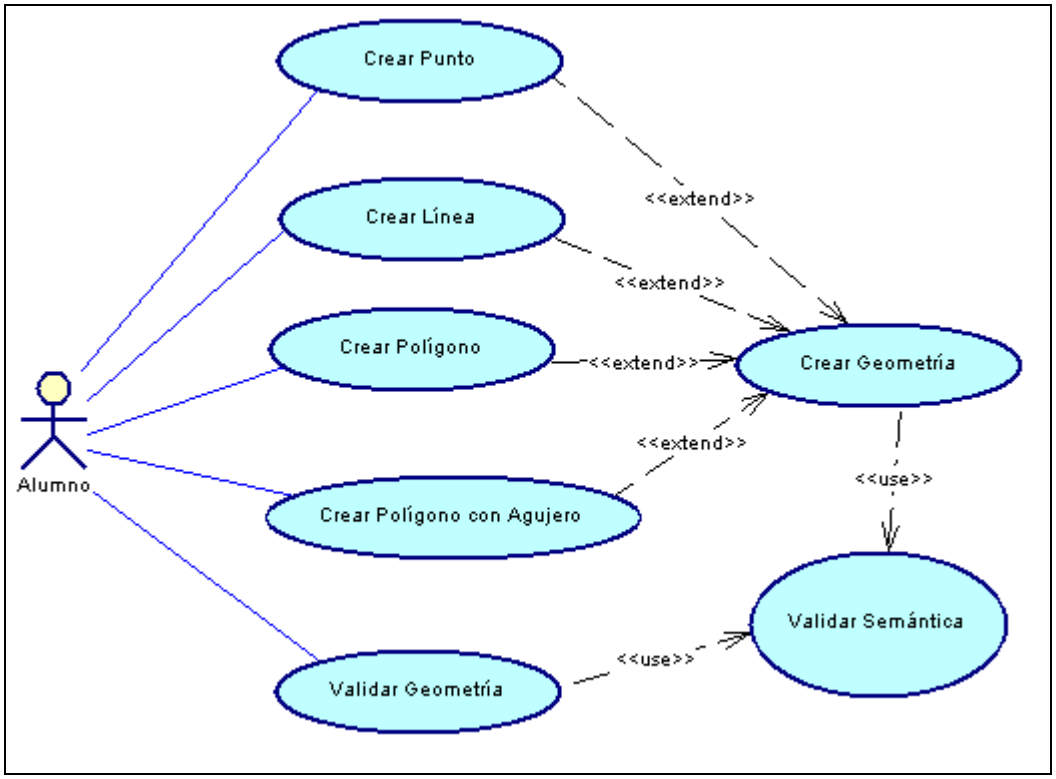



*Figura 132. Diagrama de casos de uso para el usuario alumno*

### 4.1.3.3 Descripción de los casos de uso

| **CU-1** | **Crear Capa** | |
|---|---|---|
| Actores: | Profesor | |
| Descripción: | Da de alta una capa creando la tabla con la columna SDO_GEOMETRY correspondiente así como los metadatos asociados a la misma. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe tener el rol profesor. | |
| Poscondiciones: | Capa creada y disponible para la inserción de geometrías. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de creación de capas |
| | 0.2 | Cumplimenta los datos solicitados en la pantalla de introducción de datos. |
| | 0.3 | Selecciona la acción guardar capa. |
| | 0.4 | Comprobación de capa o metadatos existentes |
| | 0.5 | Creación de la tabla correspondiente a la capa |
| | 0.6 | Inserción de los metadatos de la capa |
| | 0.7 | Inserción de la referencia a la capa en la tabla catálogo de capas disponibles en el sistema, para guardar las geometrías que se creen desde la aplicación. |
| Excepciones: | Paso | Descripción: |
| | 0.3 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de |

| | | |
|---|---|---|
| | | datos, longitudes…, si se detecta algún error en los mismos, se avisa al usuario para que introduzca las correcciones necesarias. |
| | 0.4 | Si ya existe una tabla con el nombre indicado por el usuario o ya existen los metadatos de una capa nombrada del mismo modo, se avisa al usuario para que introduzca un nombre de capa distinto. |

*Tabla 12. Caso de Uso CU-1*

| **CU-2** | **Crear Punto** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Da de alta una nueva geometría de tipo punto, insertándola en la capa seleccionada por el usuario. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe existir una capa en la que insertar la geometría. | |
| Poscondiciones: | Geometría insertada en la capa correspondiente y presentación de la sentencia SQL utilizada para realizar la inserción, así como de una tabla explicativa de la misma. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de creación de geometrías |
| | 0.2 | Selecciona la capa en la que se va a insertar la geometría |
| | 0.3 | Seleccionar la acción Crear Punto |
| | 0.4 | Cumplimenta los datos solicitados en la pantalla de introducción de datos. |
| | 0.5 | Selecciona la acción guardar geometría. |
| Excepciones: | Paso | Descripción: |
| | 0.5 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de datos, longitudes…, si se detecta algún error en los mismos, se avisa al usuario para que introduzca las correcciones necesarias. |
| Casos de los que depende: | CU-6 Crear Geometría | |
| Notas: | Los pasos recogidos en el flujo, no están completos, el resto se detallan en el caso de uso CU-6 Crear Geometría del que extiende éste. | |

*Tabla 13. Caso de Uso CU-2*

| **CU-3** | **Crear Línea** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Da de alta una nueva geometría de tipo línea, insertándola en la capa seleccionada por el usuario. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe existir una capa en la que insertar la geometría. | |
| Poscondiciones: | Geometría insertada en la capa correspondiente y presentación de la sentencia SQL utilizada para realizar la inserción, así como de una tabla explicativa de la misma. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de creación de geometrías |
| | 0.2 | Selecciona la capa en la que se va a insertar la geometría |
| | 0.3 | Seleccionar la acción Crear Línea |
| | 0.4 | Cumplimenta los datos solicitados en la pantalla de introducción de datos. |
| | 0.5 | Selecciona la acción guardar geometría. |
| Excepciones: | Paso | Descripción: |
| | 0.5 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de datos, longitudes…, si se detecta algún error en los mismos, |

| | | |
|---|---|---|
| | | se avisa al usuario para que introduzca las correcciones necesarias. |
| Casos de los que depende: | CU-6 Crear Geometría | |
| Notas: | Los pasos recogidos en el flujo, no están completos, el resto se detallan en el caso de uso CU-6 Crear Geometría del que extiende éste. | |

*Tabla 14. Caso de Uso CU-3*

| **CU-4** | **Crear Polígono** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Da de alta una nueva geometría de tipo polígono, insertándola en la capa seleccionada por el usuario. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe existir una capa en la que insertar la geometría. | |
| Poscondiciones: | Geometría insertada en la capa correspondiente y presentación de la sentencia SQL utilizada para realizar la inserción, así como de una tabla explicativa de la misma. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de creación de geometrías |
| | 0.2 | Selecciona la capa en la que se va a insertar la geometría |
| | 0.3 | Seleccionar la acción Crear Polígono |
| | 0.4 | Cumplimenta los datos solicitados en la pantalla de introducción de datos. |
| | 0.5 | Selecciona la acción guardar geometría. |
| Excepciones: | Paso | Descripción: |
| | 0.5 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de datos, longitudes…, si se detecta algún error en los mismos, se avisa al usuario para que introduzca las correcciones necesarias. |
| Casos de los que depende: | CU-6 Crear Geometría | |
| Notas: | Los pasos recogidos en el flujo, no están completos, el resto se detallan en el caso de uso CU-6 Crear Geometría del que extiende éste. | |

*Tabla 15. Caso de Uso CU-4*

| **CU-5** | **Crear Polígono con Agujero** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Da de alta una nueva geometría de tipo polígono, insertándola en la capa seleccionada por el usuario. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe existir una capa en la que insertar la geometría. | |
| Poscondiciones: | Geometría insertada en la capa correspondiente y presentación de la sentencia SQL utilizada para realizar la inserción, así como de una tabla explicativa de la misma. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de creación de geometrías |
| | 0.2 | Selecciona la capa en la que se va a insertar la geometría |
| | 0.3 | Seleccionar la acción Crear Polígono con Agujero |
| | 0.4 | Cumplimenta los datos de la geometría externa solicitados en la pantalla de introducción de datos. |
| | 0.5 | Cumplimenta los datos de la geometría interna solicitados en la pantalla de introducción de datos |
| | 0.6 | Selecciona la acción guardar geometría. |
| Excepciones: | Paso | Descripción: |
| | 0.6 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de |

| | | |
|---|---|---|
| | | datos, longitudes…, si se detecta algún error en los mismos, se avisa al usuario para que introduzca las correcciones necesarias. |
| Casos de los que depende: | CU-6 Crear Geometría | |
| Notas: | Los pasos recogidos en el flujo, no están completos, el resto se detallan en el caso de uso CU-6 Crear Geometría del que extiende éste. | |

*Tabla 16. Caso de Uso CU-5*

| **CU-6** | **Crear Geometría** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Da de alta una nueva geometría. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>Debe existir una capa en la que insertar la geometría.<br>Debe haber cumplimentado los datos del tipo de geometría a insertar | |
| Poscondiciones: | Geometría insertada | |
| Flujo: | Paso | Descripción: |
| | 0.1 | Composición del constructor SQL de la geometría a partir de los datos cumplimentados por el usuario. |
| | 0.2 | Validar Semántica de la geometría |
| | 0.3 | Construcción de la sentencia de inserción de la geometría en la capa |
| | 0.4 | Ejecución de la sentencia de inserción |
| Excepciones: | Paso | Descripción: |
| | 0.4 | Se realizan la validación semántica de la geometría, si se detecta algún error, se avisa al usuario para que introduzca las correcciones necesarias. |
| Casos dependientes: | CU-2 Crear Punto, CU-3 Crear Línea, CU-4 Crear Polígono, CU-5 Crear Polígono con Agujero. | |
| Casos de los que depende | CU-8 Validar Semántica | |
| Notas: | Los casos de uso CU-2 Crear Punto, CU-3 Crear Línea, CU-4 Crear Polígono, CU-5 Crear Polígono con Agujero extienden de éste; y éste a su vez utiliza el caso de uso CU-8 Validar Semántica. | |

*Tabla 17. Caso de Uso CU-6*

| **CU-7** | **Validar Geometría** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Valida el constructor de una geometría. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO. | |
| Poscondiciones: | Resultado de la validación. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | El usuario accede a la opción de validación de geometrías |
| | 0.2 | Cumplimenta los datos de la geometría externa solicitados en la pantalla de introducción de datos |
| | 0.3 | Selecciona la acción validar |
| | 0.4 | Se realiza la validación sintáctica del constructor y parte de la validación semántica. |
| Excepciones: | Paso | Descripción: |
| | 0.3 | Se realizan las validaciones de los datos cumplimentados por el usuario, tales como campos obligatorios, tipos de datos, longitudes, etc., si se detecta algún error en los mismos, se avisa al usuario para que introduzca las correcciones necesarias. |
| | 0.4 | Se realiza la validación sintáctica del constructor y parte de la validación semántica, si se detecta algún error, se avisa al usuario para que introduzca las correcciones necesarias |

| | |
|---|---|
| Casos de los que depende | CU-8 Validar Semántica |
| Notas: | Los pasos recogidos en el flujo, no están completos, el resto se detallan en el caso de uso CU-8 Validar Semántica del que hace uso. |

*Tabla 18. Caso de Uso CU-7*

| **CU-8** | **Validar Semántica** | |
|---|---|---|
| Actores: | Profesor, Alumno | |
| Descripción: | Valida semánticamente el constructor de una geometría. | |
| Precondiciones: | El usuario debe haberse logado en el sistema ADGSO.<br>El constructor de la geometría se ha validado sintácticamente | |
| Poscondiciones: | Resultado de la validación. | |
| Flujo: | Paso | Descripción: |
| | 0.1 | Ejecución de la función de validación que proporciona Oracle para validar geometrías. |
| | 0.2 | Comunicación del número y texto del error proporcionado por Oracle |
| Excepciones: | Paso | Descripción: |
| Casos dependientes | CU-6 Crear Geometría, CU-7 Validar Geometría | |
| Notas: | CU-6 Crear Geometría, CU-7 Validar Geometría hacen uso de éste. | |

*Tabla 19. Caso de Uso CU-8*

## 4.1.4 Otros requisitos no funcionales

### 4.1.4.1 Interfaz

Se desarrollará una aplicación Web que presentará interfaces sencillas e intuitivas en lo que se refiere a la cumplimentación de datos por parte del usuario.

Para la presentación de resultados de las acciones realizadas por el usuario se utilizarán mensajes tanto en los resultados negativos como positivos.

### 4.1.4.2 Seguridad

El acceso a la aplicación se hará mediante la autenticación previa del usuario y marcará el acceso a la funcionalidad propia de su rol.

### 4.1.4.3 Ayuda

La aplicación tiene principalmente un fin didáctico por lo que la ayuda a presentar al usuario debe ser clara y concisa. Para ello desde las páginas del aplicativo se proporcionará un enlace al manual de usuario según el rol del mismo.

### 4.1.4.4 Internalización de mensajes

El aplicativo debe estar preparado para soportar la internalización de mensajes En principio los idiomas disponibles serán castellano e inglés.

#### 4.1.4.5 Accesibilidad

No se difinió como requisito inicial del proyecto contemplar el cumplimiento de las normas establecidas [AENOR, 2004] para hacer accesibles las páginas de la aplicación. Por otro lado, este requisito ha sido incorporado en fase ya avanzada del proyecto y sí se ha seguido las normas, llegando al cumplir todos los requisitos de accesibilidad a excepción de las limitaciones impuestas por el Framework de desarrollo del aplicativo.

# 4.2 Diseño

En este apartado se va a tratar una de las partes más relevantes a la hora de realizar un proyecto software. El objetivo principal es diseñar la estructura y componentes del sistema, necesarios para cumplir los requisitos establecidos en el análisis de las necesidades del aplicativo, de modo que se consiga el paso a la implementación con tan solo realizar un refinamiento del diseño.

## 4.2.1 Diseño de la arquitectura

Al tratarse de una aplicación Web, estará formada por un cliente, que será el navegador, y por un servidor de aplicaciones Web. El servidor Web enviará las páginas Web que le sean requeridas por el navegador cliente y por el propio estado del Servidor, el cual se encargará de mostrarlas al usuario para conseguir el fin de la aplicación. La información manejada por la aplicación se encuentra almacenada en una base de datos Oracle.

Es un estilo de arquitectura típico de tres capas:

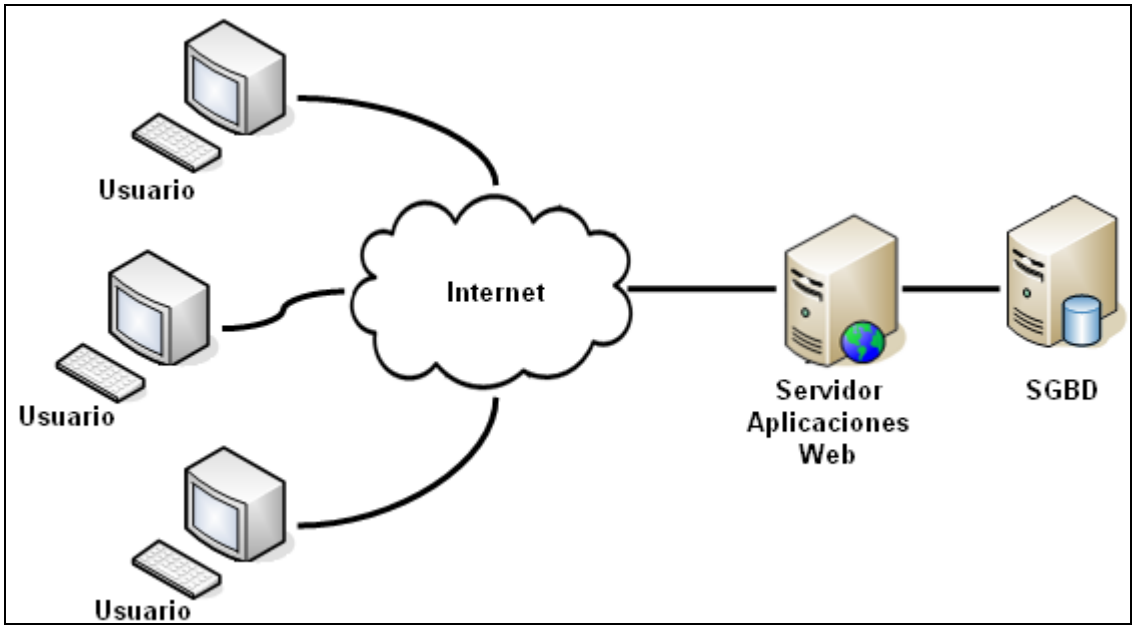



*Figura 133. Arquitectura aplicación ADGSO*

- La primera capa se encuentra del lado del cliente. Contiene todo lo necesario para que el usuario pueda visualizar e interaccionar con la aplicación. En esta capa se encontrará el navegador y las páginas Web de la aplicación.

- La segunda capa estará formada por el código fuente de la aplicación y el servidor Web. Esta es la capa que proporciona la funcionalidad.

- La tercera capa está formada por el Sistema Gestor de Base de Datos y la Base de Datos que contiene la información que se maneja en la aplicación.

## 4.2.2 Tecnologías a utilizar

### 4.2.2.1 Cliente

La aplicación debe soportar todo tipo de navegadores. Sin embargo, dos son los más importantes y los más usados en la actualidad: Internet Explorer y Mozilla Firefox. Estos dos, han sido los más utilizados a la hora de realizar la aplicación.

### 4.2.2.2 Servidor de Aplicaciones

Como servidor de aplicaciones se ha optado por Tomcat. Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Dado que fue escrito en Java, funciona en cualquier sistema operativo que disponga de una máquina virtual Java. Es fiable, de uso muy extendido en las empresas y fácil de instalar y configurar.

La versión utilizada es la 6.0.18 compatible con las versiones de servlets 2.5 y JSP 2.1.

### 4.2.2.3 Servidor de Base de datos

Dado que el proyecto está centrado en el estudio de Oracle Spatial la base de datos que va a dar soporte a la aplicación es Oracle.

La versión de la base de datos de Oracle utilizada es la 11g.

### 4.2.2.4 IDE de desarrollo

El IDE utilizado para el desarrollo es Eclipse. Es un entorno de desarrollo integrado de código abierto multiplataforma, de uso muy extendido y que proporciona gran cantidad de plugins para el desarrollo de aplicaciones, definición y manipulación de modelos de software, aplicaciones Web, etc.

La versión de Eclipse utilizada es la 3.4.1.

#### 4.2.2.5 Framework de desarrollo

Para realizar la implementación de la aplicación se ha decidido apostar por el Framework de desarrollo JSF (Java Server Faces), que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE, permitiendo la creación de aplicaciones Web de forma ágil y modulada en componentes.

La implementación utilizada es la proporcionada por Sun Microsystems, JSF 1.2.

### 4.2.3 Diseño funcional

El Framework JSF está basado en el patrón de desarrollo MVC (Modelo Vista Controlador). Este patrón separa la lógica de control (sabe que cosas hay que hacer pero no como), la lógica de negocio (sabe cómo se hacen las cosas) y la lógica de presentación (sabe como interactuar con el usuario).

La vista presenta el modelo en el formato adecuado con el que el usuario debe interactuar, es decir presenta la interfaz al usuario.

El controlador es el encargado de responder a los eventos y acciones que invoca el usuario desde la vista, invocar peticiones al modelo y retornar los datos a la vista.

El modelo es la representación específica de la información de negocio con la cual el sistema opera. Incorpora la capa del dominio y persistencia, es la encargada de guardar los datos en un medio persistente.

Este patrón de diseño es muy popular en el desarrollo de aplicaciones Web, ya la separación entre vistas, controladores y modelos, permite desarrollar productos de calidad, flexibles, más fácilmente mantenibles y escalables. Siguiendo el patrón MVC, el conjunto de componentes de la aplicación se va a dividir en tres grupos:

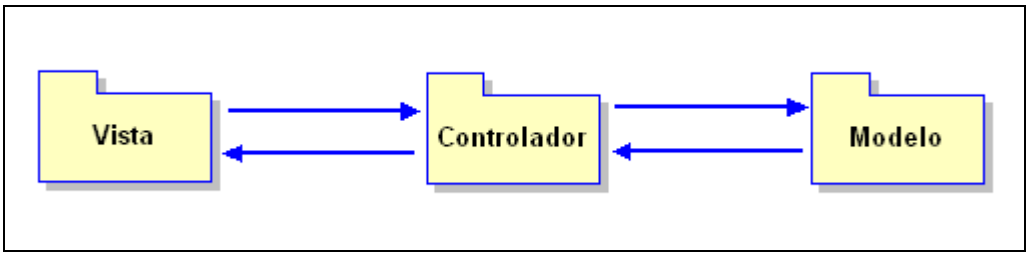



*Figura 134. Patrón de diseño aplicado a ADGSO*

En este diagrama el flujo de control está representado por las flechas, y se explica a continuación:

1. El usuario interactúa con la interfaz de alguna manera (Ej. presionando un botón, enlace).

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario.

3. El controlador accede al modelo, posiblemente actualizando los datos enviados por el usuario.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario.

5. La vista usa los datos enviados por el controlador acerca del modelo para generar la interfaz apropiada para el usuario donde refleja los cambios.

6. La interfaz espera por nuevas interacciones de usuario para iniciar nuevamente el ciclo.

A continuación se verán los elementos que componen cada uno de los grupos de la figura “*Figura 134. Patrón de diseño aplicado a ADGSO*”.

### 4.2.3.1 Vista

En el siguiente diagrama se muestra el conjunto de componentes que forman la vista de la aplicación, es decir, la interfaz de la misma:

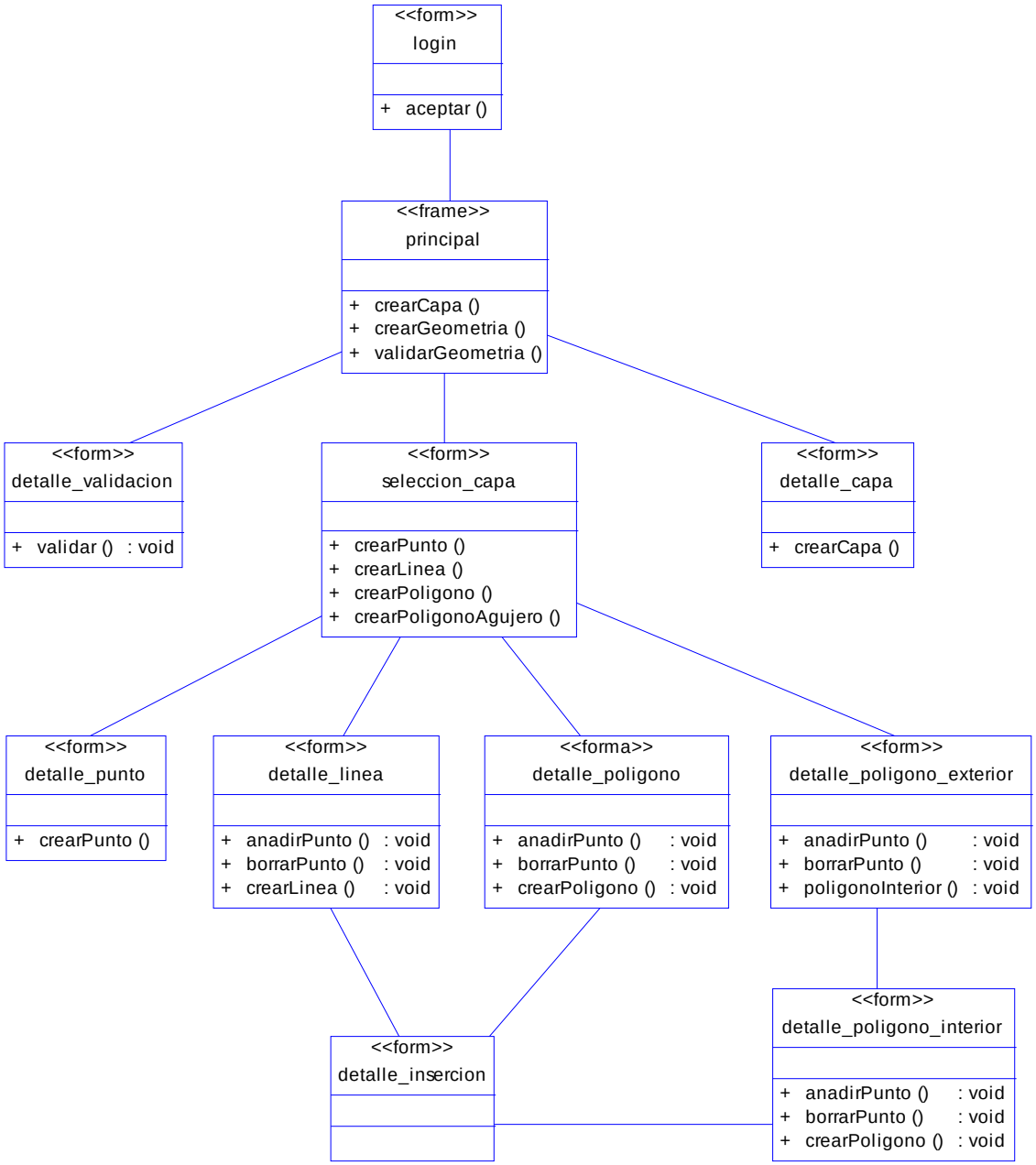



*Figura 135. Diagrama de componentes de la vista*

Los elementos que forman la vista están compuestos por frames y formularios, que se van a traducir posteriormente en páginas JSP. En el apartado "*4.2.5 Diseño de la interfaz*" de este documento se estudia más a fondo la composición de la interfaz.

### 4.2.3.2 Controlador

En el siguiente diagrama se muestra el conjunto de componentes que forman el controlador de la aplicación:

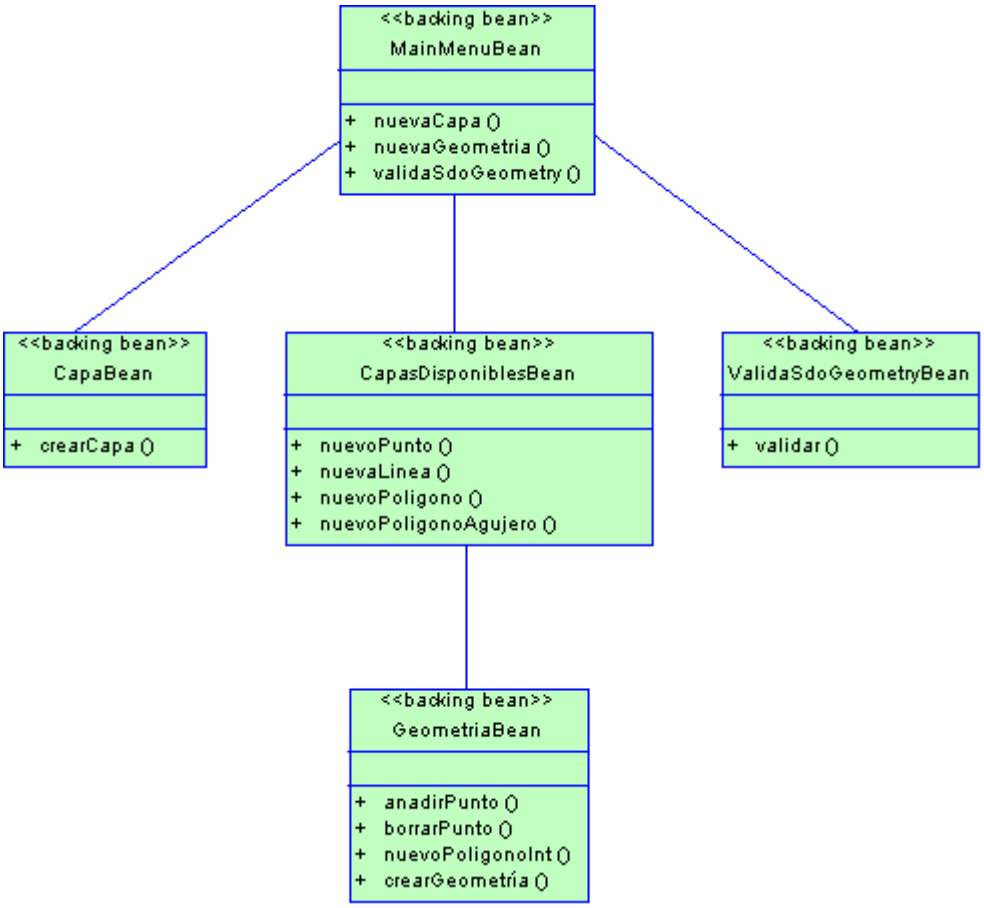



*Figura 136. Diagrama de componentes del controlador*

Los elementos que forman el controlador corresponden a lo que en JSF se conoce como beans de respaldo. Estos junto con el servlet de JSF son los encargados de atender los eventos y acciones de usuario, delegarlas a la capa del modelo y llevar el control de navegación entre las vistas.

### 4.2.3.3 Modelo

En el siguiente diagrama se muestra el conjunto de componentes que forman el modelo de la aplicación:

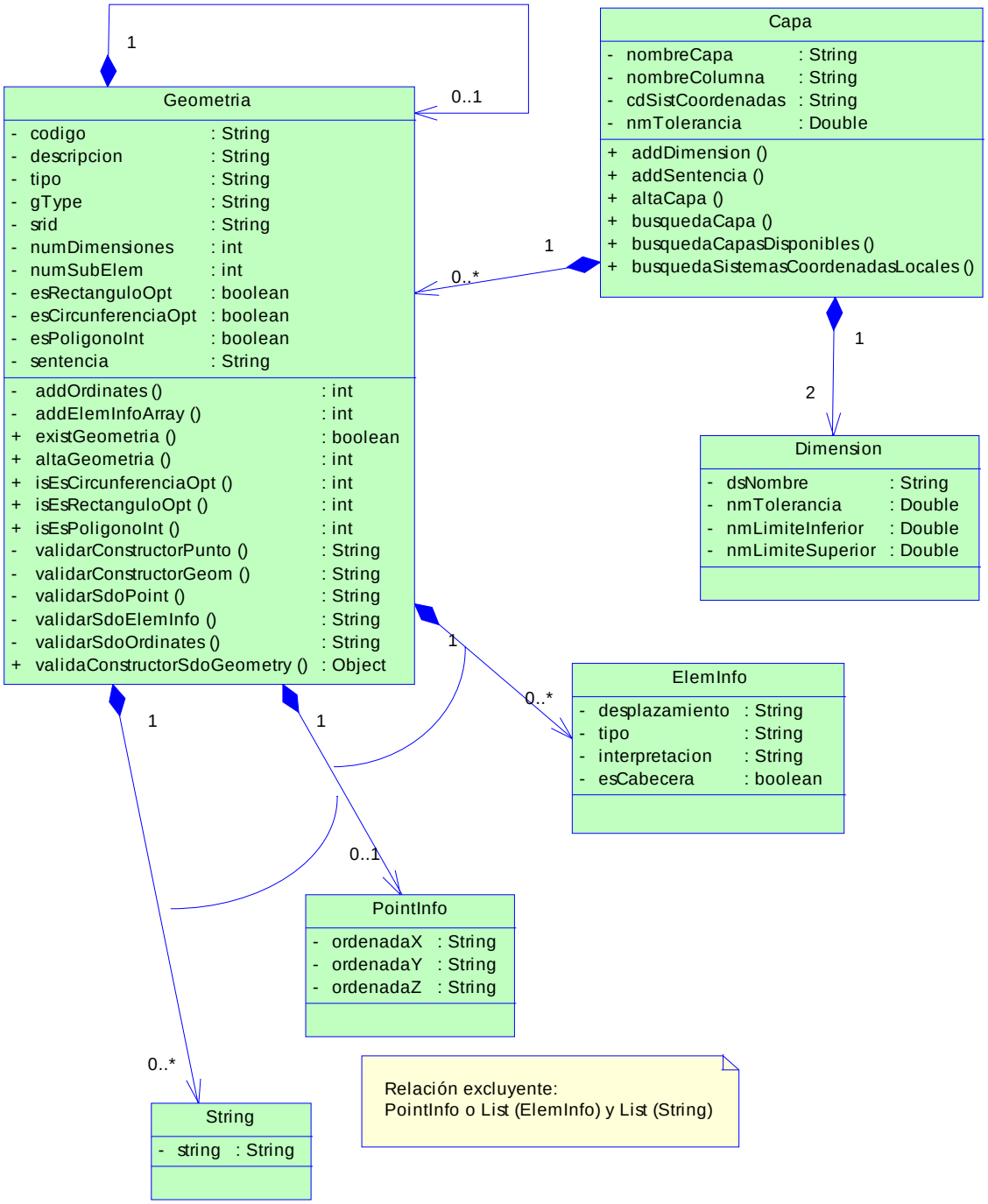



*Figura 137. Diagrama de componentes del modelo*

Estos componentes son los que contienen la información necesaria para generar las sentencias SQL necesarias para crear capas, insertar geometrías y validar constructores de geometrías; y se encargan de persistir dicha información en la base de datos.

## 4.2.4 Diseño de base de datos

El modelo de base de datos de la aplicación está compuesto por una única tabla catálogo en la que se registran las capas que se han dado de alta desde la aplicación y por las tablas correspondientes a las capas que se crean desde la propia aplicación.

### 4.2.4.1 Tabla catálogo de capas

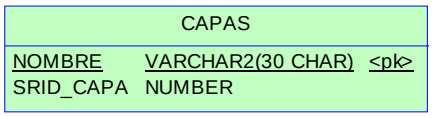



*Figura 138. Tabla catálogo capas*

### 4.2.4.2 Tablas de capas

Todas las tablas correspondientes a las capas tienen la siguiente estructura:

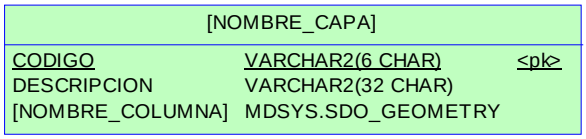



*Figura 139. Tabla correspondiente a una capa*

- [NOMBRE_CAPA]: corresponde al nombre que asigna a la capa el usuario que la da de alta desde la aplicación.
- Código: código de la geometría. Este campo es el identificador de la geometría, por lo que es clave primaria de la tabla.
- Descripción: Breve descripción de la geometría de carácter obligatorio.
- [NOMBRE_CAPA]: al dar de alta una capa el usuario también debe indicar el nombre de la columna de tipo SDO_GEOMETRY en el que se van a guardar las geometrías cuando se den de alta desde la aplicación.

La creación de capas implica la inserción de los metadatos asociados a la misma. Estos datos se persisten en la vista proporcionada por Oracle USER_SDO_GEOM_METADATA. La estructura de la vista y los campos a indicar en la misma se describieron en el apartado "*3.5 Metadatos*".

## 4.2.5 Diseño de la interfaz

A continuación se describe la interfaz que permitirá a los diferentes usuarios del sistema, acceder a las funcionalidades descritas en la especificación de requisitos del aplicativo.

La información que se observa en las siguientes figuras muestra las diferentes partes en las que se divide la interfaz, enlace, botones, cuadros de texto, etc. A continuación se detallan los elementos utilizados para generar las pantallas y el significado de cada una de ellas:

- *Text*: Texto fijo que aparece en la interfaz correspondiente.

- *Text Box*: Elemento de la interfaz que permite introducir texto simple.
- *Text Area*: Elemento de la interfaz que permite introducir múltiples líneas de texto.
- *Link*: Es un enlace a otra página Web, ya sea cliente o servidor.
- *Button*: Elemento que va a permitir realizar una acción.
- *Selection List*: Lista de opciones que se le presentan al usuario, para que selecciona una de ellas.

### 4.2.5.1 Pantalla Principal

Es la pantalla en de acceso a la aplicación a través de la cual el usuario se identifica frente al sistema.

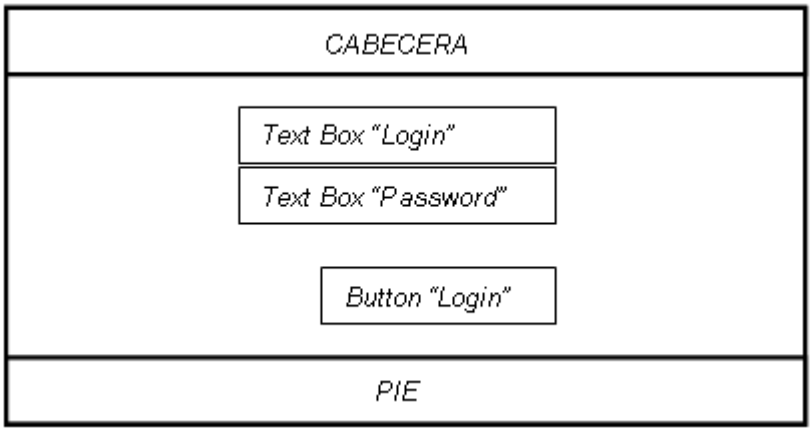



*Figura 140. Pantalla autenticación ADGSO*

Una vez autenticado, el usuario accede a la pantalla principal del la aplicación en la que se muestran las opciones que puede seleccionar según su perfil. Todas las páginas tienen la siguiente estructura:

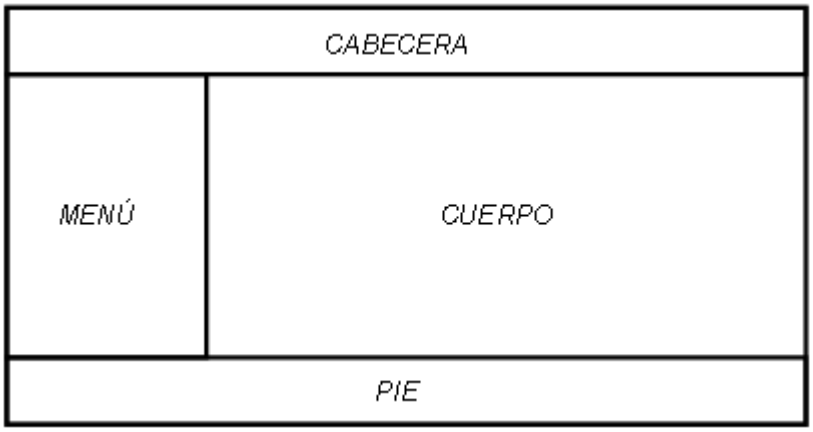



*Figura 141. Pantalla principal ADGSO*

### 4.2.5.2 Menús

Las opciones de menú difieren en función del rol del usuario de acceso. Para usuarios con rol profesor el menú muestra las siguientes opciones:


*Figura 142. Opciones menú rol Profesor*

Para los usuarios con rol Alumno:

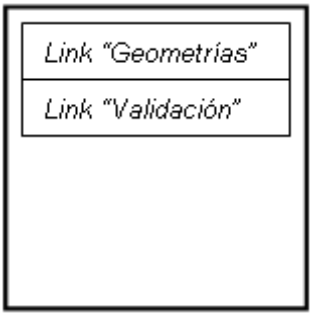



*Figura 143. Opciones menú rol Alumno*

### 4.2.5.3 Cuerpo

Tras seleccionar la opción de menú correspondiente el cuerpo muestra los controles propios de cada funcionalidad.

Todas las funcionalidades que requieran presentación de mensajes, ya sean de error o información del resultado de una operación, incorporan en la parte superior del cuerpo, justo encima de todos los controles de introducción de datos, una zona en la que se muestran dichos mensajes.

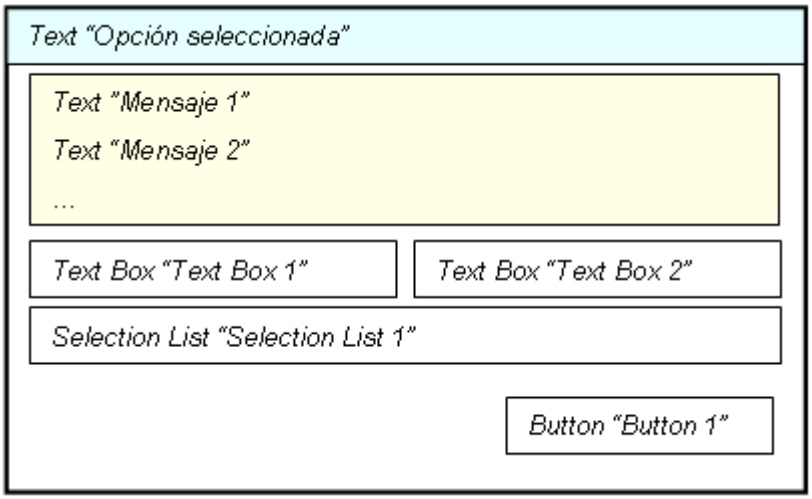



*Figura 144. Mensajes en el cuerpo*

#### 4.2.5.3.1 Cuerpo Crear Capa

La pantalla consiste en un formulario donde el usuario indicará todos los datos necesarios para dar de alta una capa. Se muestra al seleccionar la opción de menú 'Capas'.

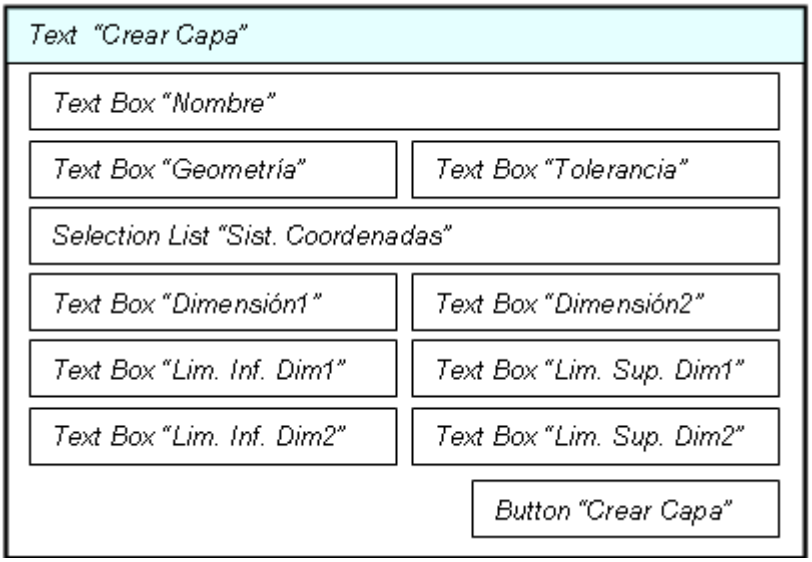



*Figura 145. Cuerpo Crear Capa*

Tras cumplimentar los datos correctamente y pulsar el botón 'Crear Capa', se muestra el resultado de la operación y la lista de sentencias ejecutadas para crear la nueva capa.

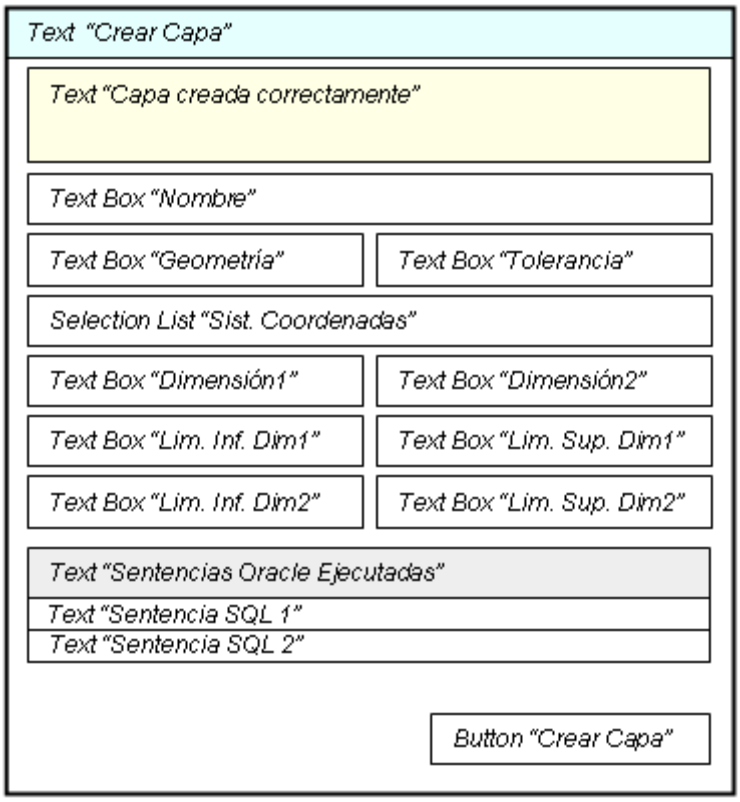



*Figura 146. Cuerpo Resultado Crear Capa*

#### 4.2.5.3.2 Cuerpo Selección Capa

La pantalla que se muestra al seleccionar la opción de menú 'Geometrías' consiste en un formulario donde el usuario selecciona la capa en la que va a insertar la geometría.

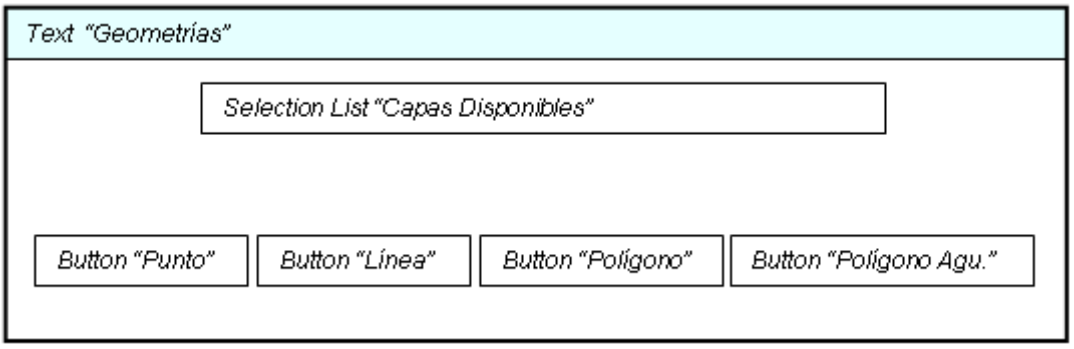



*Figura 147. Cuerpo Selección Capa*

Tras seleccionar la capa en la que insertar la geometría se pulsa el botón correspondiente al tipo de geometría que se quiera crear.

#### 4.2.5.3.3 Cuerpo Crear Punto

La pantalla consiste en un formulario donde el usuario indicará todos los datos necesarios para dar de alta una geometría correspondiente a un punto.

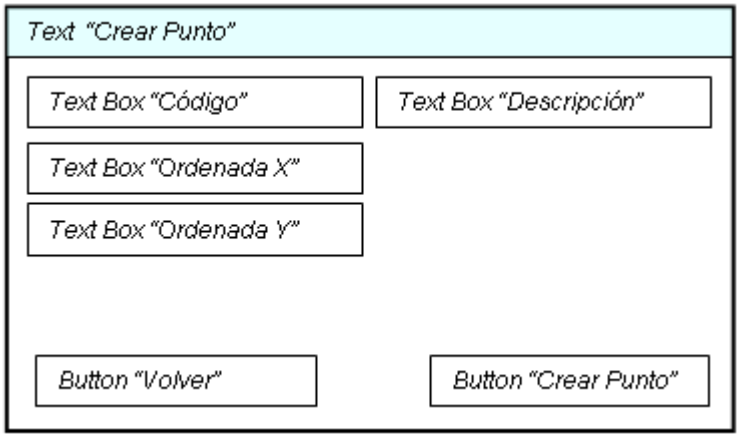



*Figura 148. Cuerpo Crear Punto*

Tras cumplimentar los datos correctamente y pulsar el botón 'Crear Punto', se muestra el resultado de la operación y la sentencia ejecutada para crear el punto.

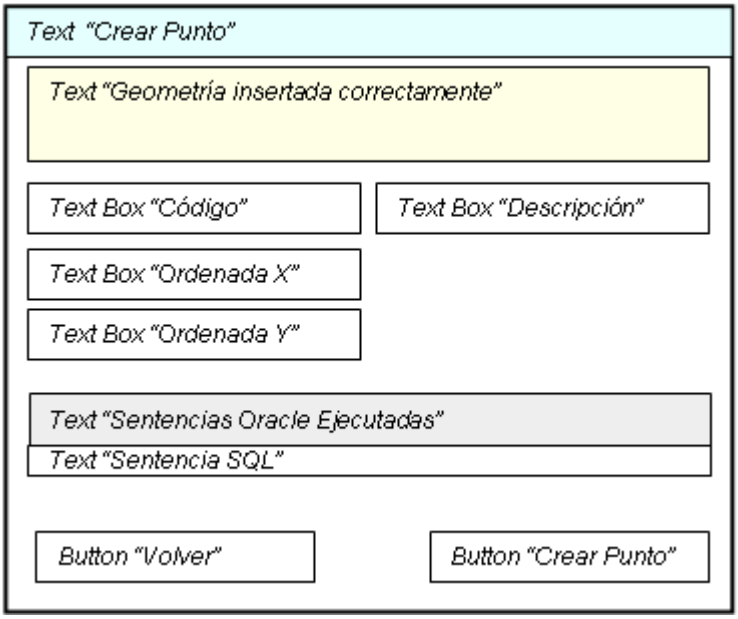



*Figura 149. Cuerpo Resultado Crear Punto*

#### 4.2.5.3.4 Cuerpo Crear Línea

La pantalla consiste en un formulario donde el usuario indicará todos los datos necesarios para dar de alta una geometría correspondiente a una línea.

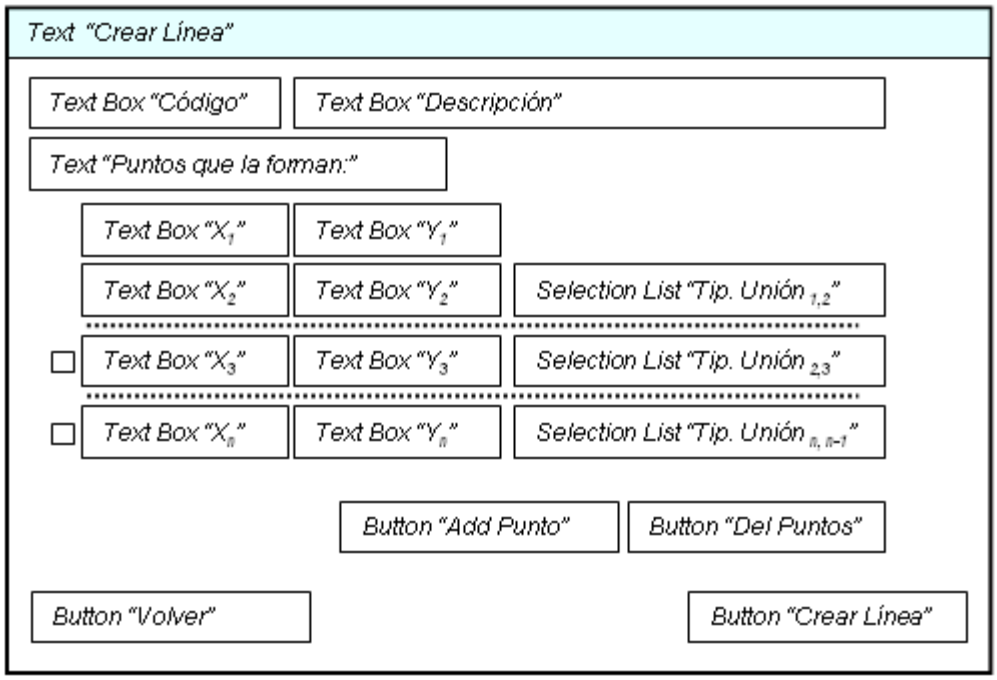



*Figura 150. Cuerpo Crear Línea*

Tras cumplimentar los datos correctamente y pulsar el botón 'Crear Línea', se muestra la pantalla de detalle del resultado de la operación, que contiene la sentencia ejecutada para crear la línea y una tabla explicativa de la misma.

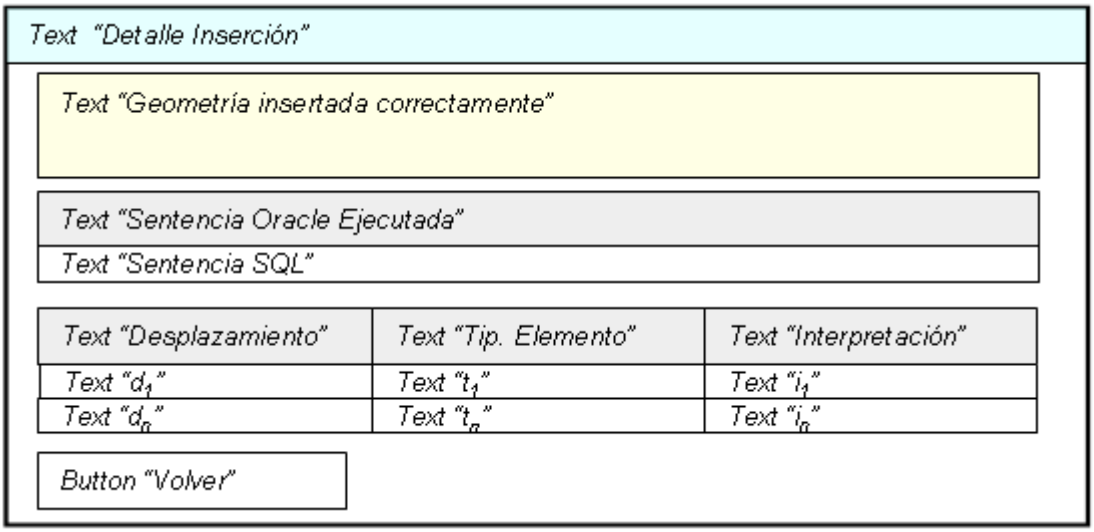



*Figura 151. Cuerpo Detalle Inserción*

#### 4.2.5.3.5 Cuerpo Crear Polígono

La pantalla consiste en un formulario donde el usuario indicará todos los datos necesarios para dar de alta una geometría correspondiente a un polígono.

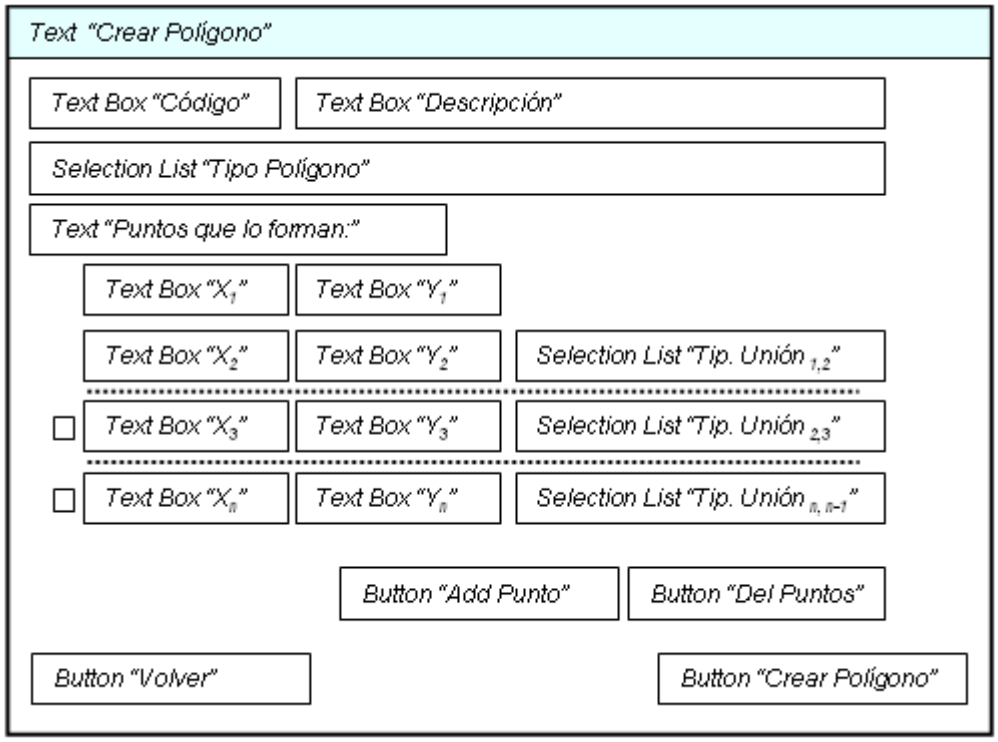



*Figura 152. Cuerpo Crear Polígono*

En el caso de que el tipo de polígono sea un rectángulo o circunferencia optimizada el número de puntos se fija a dos y tres respectivamente sin dar la posibilidad de añadir o eliminar puntos.

Tras cumplimentar los datos correctamente y pulsar el botón 'Crear Polígono', se muestra la pantalla de detalle del resultado de la operación, que contiene la sentencia ejecutada para crear el polígono y una tabla explicativa de la misma (Véase la figura "*Figura 151. Cuerpo Detalle Inserción*")

#### 4.2.5.3.6 Cuerpo Crear Polígono con Agujero

La primera pantalla consiste en un formulario donde el usuario indicará todos los datos referentes al polígono externo. Tiene el mismo aspecto que el de la figura "*Figura 152. Cuerpo Crear Polígono*", difiere en que el botón 'Crear Polígono' en este caso es un botón 'Siguiente' para dar paso a la pantalla en la que se deben indicar los datos del polígono interno. En esta segunda pantalla es donde se muestra e. botón 'Crear Polígono'.

Tras cumplimentar los datos correctamente del polígono externo en la primera pantalla y el polígono interno en la segunda pantalla y pulsar el botón 'Crear Polígono', se muestra la pantalla de detalle del resultado de la operación, que contiene la sentencia ejecutada para crear el polígono con agujero y una tabla explicativa de la misma (Véase la figura "*Figura 151. Cuerpo Detalle Inserción*").

#### 4.2.5.3.7 Cuerpo Validar Geometría

La pantalla consiste en un formulario donde el usuario indicará todos los datos necesarios para validar el constructor de una geometría escrito por él. Se muestra al seleccionar la opción de menú 'Validar'.

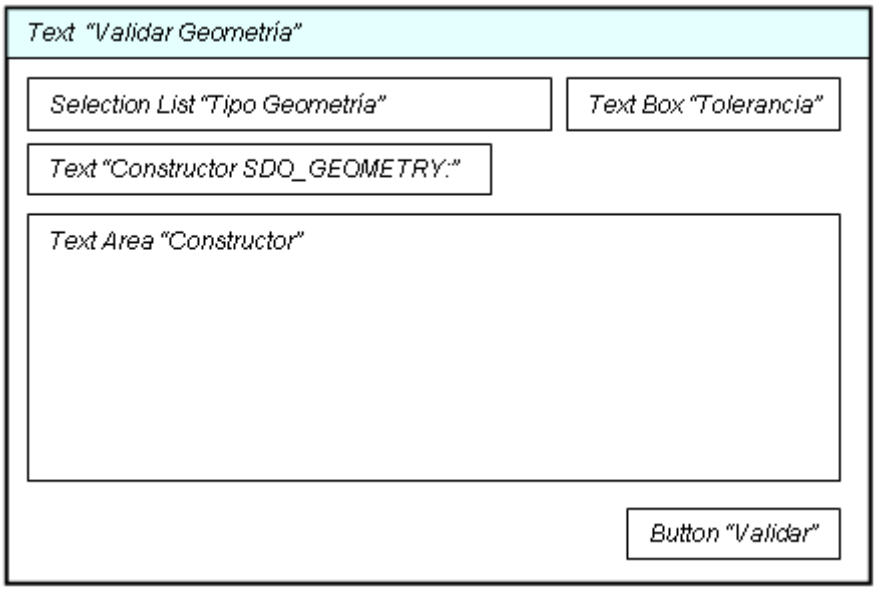



*Figura 153. Cuerpo Validar Geometría*

Tras cumplimentar los datos correctamente y pulsar el botón 'Validar', se muestra el resultado de la validación. Si la geometría no es correcta se muestra el error en el apartado de mensajes, como en el resto de pantallas, mientras que si la geometría es correcta se muestra un mensaje informando al usuario y una tabla explicativa correspondiente al constructor indicado.

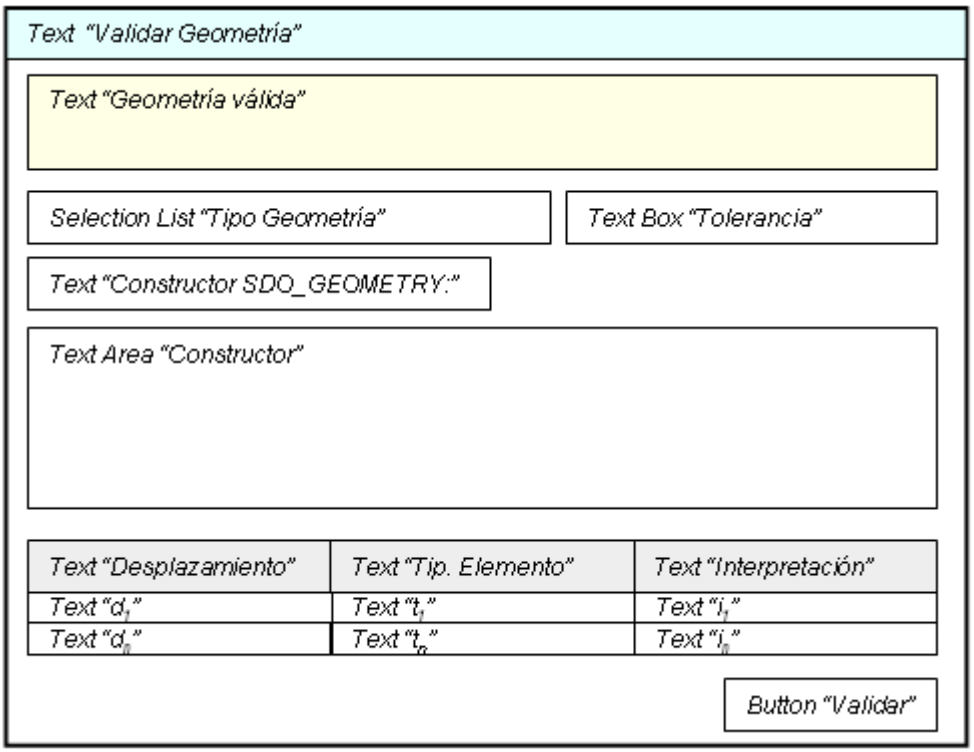



*Figura 154. Cuerpo Resultado Validar Geometría*

# 4.3 Implementación

Este apartado tiene como finalidad mostrar el proceso de implementación de la aplicación Web ADGSO.

En primer lugar se va a dar una breve descripción de las herramientas y técnicas de programación utilizadas, algunas de las cuales ya se han introducido en apartados anteriores:

- Para la edición del código fuente se ha utilizado el IDE de desarrollo Eclipse sobre plataforma Windows. Incluye las herramientas necesarias para compilar y depurar el código, por lo que no ha sido necesario utilizar ningún compilador o depurador externo. Así mismo el despliegue de la aplicación se ha realizado mediante Apache Ant, desde el mismo desarrollo, utilizando un plugin de Eclipse.

- La aplicación se ha desarrollado en Java haciendo uso del framework JSF. Las páginas de la aplicación se han generado haciendo uso del sistemas de plantillas Facelets y de los componentes de JSF en las páginas JSP.

- La base de datos utilizada es Oracle, en su versión Express Edition que proporciona todos los recursos usados en ADGSO y a la vez ofrece una instalación ligera del producto de Oracle.

- El despliegue de la aplicación se ha realizado sobre el contenedor de servlets Apache Tomcat. Que además ha proporcionado la seguridad a nivel de contenedor. Es uno de los mecanismos más populares y menos costosos, ya que se aplica directamente el sistema veterano y conocido de securización de aplicaciones Web. Los roles y usuarios se definen en el fichero de configuración del Tomcat *tomcat-users.xml*.

- Para la parte del cliente, se han utilizado principalmente los navegadores Web Internet Explorer, Mozilla Firefox.

Bajo este entorno se ha desarrollado el aplicativo ADGSO siguiendo las pautas que se describen en los siguientes apartados.

## 4.3.1 Programación del Código

Para la programación del código fuente, se parte de la información extraída en la fase de diseño.

En un primer lugar, basado en el diseño de interfaz detallado en el apartado "*4.2.5 Diseño de la interfaz*" se crea una maqueta para disponer del interfaz a presentar al usuario. La interfaz ha sufrido pequeños retoques hasta disponer del interfaz definitivo.

Se han seguido las indicaciones del W3C (World Wide Web Consortium) en lo que a realización de páginas se refiere. Por esto, toda la apariencia de la página se encuentra encapsulada dentro de una hoja de estilo o *CSS*. De este modo, se consigue una separación entre contenidos y apariencia.

En cuanto al cumplimiento de normas de accesibilidad, se han seguido las normas básicas, incluyendo no añadir javascript intrusivo en las páginas, pero los componentes JSF utilizados para crear las páginas JSP incluyen javascript intrusivo en las páginas por lo que la aplicación no cumple las normas de accesibilidad AA del W3C.

Una vez definida la interfaz definitiva, partiendo de las opciones de menú principales, se ha desarrollado la funcionalidad en el lenguaje de programación java, bajo el Framework JSF siguiendo el patrón MCV tal y como se indicaba en el diseño de la aplicación.

## 4.3.2 Estructuración del código fuente

La estructuración del código fuente se ha realizado siguiendo las recomendaciones para proyectos Web que utilizan tecnologías Web J2EE y que van a ser distribuidas mediante la construcción de un fichero WAR.

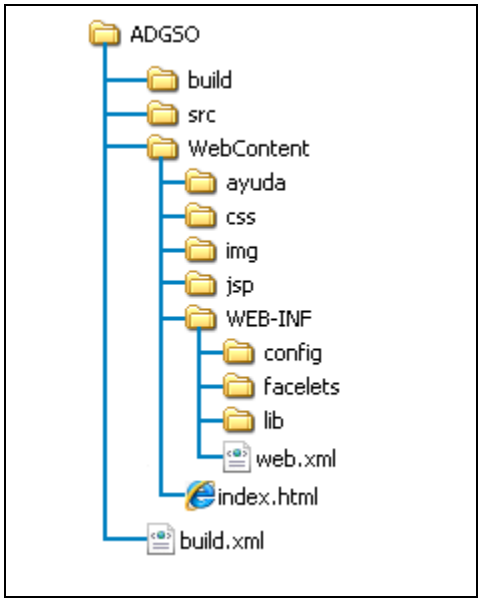



*Figura 155. Estructura código fuente ADGSO*

A continuación se detallan cada uno de los elementos de la estructura de desarrollo del proyecto:

- build: contiene los fuentes compilados y el fichero de despliegue de la aplicación ADGSO.war.

- src: contiene el código fuente de la aplicación, los ficheros de idiomas y otros ficheros de propiedades que contienen información del aplicativo tal como el fichero de propiedades de la aplicación o el nivel de logado de log4j. El

código fuente se distribuye en paquetes en base a la separación de tipos de elementos, modelo, controlador y vista.

- Webcontent: este directorio contiene las carpetas y archivos que componen el archivo WAR de despliegue de la aplicación.
    - ayuda: contiene los ficheros PDF que se visualizan desde la aplicación al seleccionar la opción de Ayuda.
    - css: directorio de hojas de estilo de aplicación.
    - img: imágenes de la aplicación, tales como fondos, logos, iconos etc.
    - jsp: contiene las páginas JSP que conforman el interfaz de la aplicación.
    - WEB-INF: contiene el descriptor de despliegue web.xml, los archivos de configuración tales como el faces-config.xml, las plantillas de Facelets y las librerías externas utilizadas por la aplicación.
    - index.html: página de inicio de la aplicación.
- build.xml: fichero de ANT que contiene las tareas necesarias para compilar y construir el fichero WAR de despliegue de la aplicación. Contiene otras tareas usadas durante el desarrollo, tales como la limpieza y despliegue en el servidor Tomcat.

# 4.4 Evaluación

En este último apartado de la documentación sobre la aplicación ADGSO, se definen las pruebas realizadas para comprobar la existencia de errores de programación.

La prueba de los programas es la técnica de confirmación de los sistemas más utilizada. Es la parte del proceso de confirmación que suele realizarse durante la aplicación y también, de una forma distinta, cuando ésta ha terminado. La prueba consiste en ejercitar el programa utilizando datos similares a los datos reales que habrán de ser ejecutados por el programa, observar los resultados y deducir la existencia de errores o insuficiencias del programa a partir de las anomalías de ese resultado.

Es muy importante comprender que la prueba nunca demuestra que un programa es correcto. Siempre es posible que existan errores aún después de la prueba más completa. La prueba de programas sólo puede demostrar la presencia de errores en un programa, no su ausencia. Por lo tanto, se considera que la prueba acertada es aquella que establece la presencia de uno o más errores en el software objeto de la prueba.

### 4.4.1 Plan de pruebas

El plan de pruebas que se va a llevar a cabo divide las pruebas en los siguientes niveles:

- Pruebas de componentes: En este tipo de pruebas se prueba el funcionamiento de cada uno de los módulos que componen el sistema total. Este tipo de pruebas se realizan en la fase de implementación probando que cada uno de los componentes hace lo que tiene que hacer antes de integrarlos en subsistemas más complejos donde la detección de errores puede ser más complicada.

- Pruebas de interfaz: En este tipo de pruebas se realizan pruebas de utilización de la interfaz probando que cada uno de los elementos (botones, cajas de texto, etc.) funciona de forma adecuada.

- Pruebas de funcionalidad: En este tipo de pruebas se comprueba la funcionalidad de la herramienta y han de ser guiadas por los casos de uso. Es decir, se prueba que el comportamiento de la herramienta en los escenarios que describen los casos de uso es el deseado.

- Pruebas de integración: En este tipo de pruebas se comprueba el funcionamiento del sistema como un todo, comprobando que las interacciones entre los distintos componentes se hace de forma adecuada. La forma de hacer la integración de los distintos módulos fue guiada por los casos de uso. En cada paso se añadían al sistema total el conjunto de módulos que implementaban alguno de los casos de uso. Así hasta completar el sistema y comprobar que la integración es satisfactoria.

- Pruebas de regresión: En este tipo de pruebas, se comprueban todos los módulos que hayan podido ser afectados tras un cambio. Durante la fase de implementación, se controlaron los cambios en el sistema probando posteriormente los módulos que hubieran sido afectados por tales cambios.

### 4.4.2 Casos de prueba

| **CP-1** | **Entrada de un usuario al sistema** |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la autenticación de usuario en el sistema. |
| Procedimiento de Prueba | El usuario en la pantalla de autenticación de usuario, introduce su Usuario y Contraseña y pulsa el botón Login. |
| Salida Esperada | Se muestra la página principal de la aplicación con las opciones de menú propias del rol del usuario. |
| Salida Obtenida | Se muestra la página principal de la aplicación con las opciones de menú propias del rol del usuario. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 20. Caso de Prueba CP-1*

| CP-2 | Entrada incorrecta de un usuario al sistema |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la autenticación incorrecta de un usuario en el sistema. |
| Procedimiento de Prueba | El usuario en la pantalla de autenticación de usuario, introduce su Usuario y Contraseña y pulsa el botón Login. |
| Salida Esperada | No se permite el acceso a la página principal de la aplicación y se muestra un mensaje informativo al usuario indicando que el Usuario y/o Login son incorrectos. |
| Salida Obtenida | No se permite el acceso a la página principal de la aplicación y se muestra un mensaje informativo al usuario indicando que el Usuario y/o Login son incorrectos. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 21. Caso de Prueba CP-2*

| CP-3 | Crear Capa |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la creación de una capa. |
| Procedimiento de Prueba | Una vez identificado como profesor, el usuario debe seleccionar la opción Capas, rellenar los datos de la misma y pulsar el botón Crear Capa. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea la tabla correspondiente a la capa, se insertan los metadatos correspondientes a la misma y se añade al catálogo de capas disponibles. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente, se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea la tabla correspondiente a la capa, se insertan los metadatos correspondientes a la misma y se añade al catálogo de capas disponibles. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 22. Caso de Prueba CP-3*

| CP-4 | Crear Punto |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la creación de un punto. |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar la opción Geometrías, seleccionar la capa en la introducir la geometría y pulsar el botón Crear Punto, rellenar los datos del punto y pulsar el botón Crear Punto. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo punto y se muestra la sentencia SQL ejecutada para la inserción de la misma. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo punto y se muestra la sentencia SQL ejecutada para la inserción de la misma. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 23. Caso de Prueba CP-4*

| **CP-5** | **Crear Línea** |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la creación de una línea. |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar la opción Geometrías, seleccionar la capa en la introducir la geometría y pulsar el botón Crear Línea, rellenar los datos de la línea y pulsar el botón Crear Línea. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo línea y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo línea y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 24. Caso de Prueba CP-5*

| **CP-6** | **Crear Polígono** |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la creación de un polígono. |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar la opción Geometrías, seleccionar la capa en la introducir la geometría y pulsar el botón Crear Polígono, rellenar los datos del polígono y pulsar el botón Crear Polígono. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo polígono y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo polígono y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 25. Caso de Prueba CP-6*

| **CP-7** | **Crear Polígono con Agujero** |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de la creación de un polígono con agujero. |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar la opción Geometrías, seleccionar la capa en la introducir la geometría y pulsar el botón Crear Polígono con Agujero, rellenar los datos del polígono externo, pulsar el botón Siguiente para cumplimentar los datos de polígono interno y pulsar el botón Crear Polígono. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo |

| | |
|---|---|
| | polígono y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, duplicados, numéricos, longitudes…), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se crea una geometría de tipo polígono y se muestra la sentencia SQL ejecutada para la inserción y la tabla explicativa de la misma. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 26. Caso de Prueba CP-7*

| **CP-8** | **Validar Geometría** |
|---|---|
| Tipo de prueba | Funcional + Interfaz |
| Objetivo | Verificar el funcionamiento de validación del constructor de una geometría. |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar la opción Validar, rellenar los datos del constructor del tipo de geometría a validar y pulsar el botón Validar. |
| Salida Esperada | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, longitudes, constructor sintácticamente correcto y correspondiente al tipo de geometría indicado), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se valida que el constructor semánticamente y se muestra un mensaje al usuario indicando que es correcto si es válido o mostrando el error Oracle correspondiente en caso de que no lo sea. |
| Salida Obtenida | Si el usuario no ha cumplimentado todos los datos correctamente (obligatorios, longitudes, constructor sintácticamente correcto y correspondiente al tipo de geometría indicado), se muestra un mensaje informativo para que realice las correcciones oportunas. Cuando los datos sean correctos, se valida que el constructor semánticamente y se muestra un mensaje al usuario indicando que es correcto si es válido o mostrando el error Oracle correspondiente en caso de que no lo sea. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 27. Caso de Prueba CP-8*

| **CP-9** | **Multiidioma** |
|---|---|
| Tipo de prueba | Integración + Interfaz |
| Objetivo | Verificar el funcionamiento del multiidioma |
| Procedimiento de Prueba | Una vez identificado como profesor o alumno, el usuario debe seleccionar el cambio de idioma de español (por defecto) a inglés o viceversa. |
| Salida Esperada | Las etiquetas y mensajes de la aplicación se muestran en el idioma correspondiente. |
| Salida Obtenida | Las etiquetas y mensajes de la aplicación se muestran en el idioma correspondiente. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 28. Caso de Prueba CP-9*

| **CP-10** | **Verificación navegadores** |
|---|---|
| Tipo de prueba | Integración + Interfaz |
| Objetivo | Verificar el funcionamiento de la aplicación en los navegadores Internet Explorer y Mozilla Firefox. |
| Procedimiento de Prueba | Probar el funcionamiento de la interfaz en los navegadores Internet Explorer y Mozilla Firefox. |
| Salida Esperada | Se ha de comprobar que la aplicación funciona correctamente en |

| | |
|---|---|
| | cualquiera de los casos. |
| Salida Obtenida | Se ha de comprobar que la aplicación funciona correctamente en cualquiera de los casos. |
| Resultado de la prueba | La prueba no ha detectado errores en la aplicación. |

*Tabla 29. Caso de Prueba CP-10*

# Capítulo 5

# Documentación

En este apartado se incluyen los manuales a distribuir junto con la aplicación ADGSO:

- Manual de Administración.
- Manuales de Usuario

## 5.1 Manual de Administración

### 5.1.1 Introducción

El presente documento va dirigido al usuario administrador del sistema para informar de los procesos para la correcta instalación y administración de la aplicación ADGSO.

Los pasos a realizar son los siguientes:

- Proceso de ejecución de scripts del modelo de datos.
- Proceso de instalación – la aplicación se despliega haciendo uso del fichero WAR.

- Ejecución de la instalación – se prueba que la instalación se ha realizado correctamente.

## 5.1.2 Proceso de instalación

### 5.1.2.1 Inventario de componentes software

Será necesario disponer de la carpeta que contiene los ficheros de la entrega del proyecto, a partir de ahora, `${ADGSO}`:

- adgso.zip – Proyecto eclipse correspondiente al código fuente de la aplicación.
- adgso.war – Ficheros de despliegue de la aplicación.
- adgso.sql – Scripts de base de datos.

### 5.1.2.2 Requisitos de hardware y software

Se debe disponer de los siguientes servicios:

- Contenedor Web/Servlets: Tomcat 6.0.20 ©. Al directorio en el que está instalado se le va a denominar a partir de ahora `${TOMCAT}`.
- SGBD: Oracle © 10g o superior.
- Herramienta de administración de bases de datos, SQL Plus, TOAD, etc.
- Navegador Web de la lista que figura a continuación:
    - Microsoft Internet
    - Mozilla Firefox

### 5.1.2.3 Procedimientos de instalación

Se deben realizar los siguientes pasos:

1. Instalación de Base de Datos: Para ello desde la herramienta de administración de bases de datos disponibles se ejecuta el script `${ADGSO}/adgso.sql`.
2. Instalación de la Aplicación: La aplicación se encuentra comprimida en el archivo 'adgso.war'. Para desplegarlo se sitúa en el directorio `${TOMCAT}\webapps` del mismo. Seguidamente se arranca el servidor y automáticamente se desplegará en ese mismo directorio en una carpeta nombrada del mismo modo que el fichero war correspondiente, en este caso `${TOMCAT}\webapps\adgso`.

3. Si la aplicación ya se hubiese desplegado con anterioridad en ese mismo servidor, se deben eliminar las carpetas `${TOMCAT}\webapp\adgso` y `${TOMCAT}\webapps\work\Catalina\localhost` para asegurar que las modificaciones realizadas se actualicen correctamente.

4. Actualización de la conexión: el fichero 'adgso.properties' del directorio `TOMCAT}\webapps\adgso\WEB-INF\classes` contiene los datos de conexión a la base de datos de ADGSO. Se deben actualizar los datos referentes al entorno actual. Es importante que el usuario de conexión que se indique tenga permiso para crear tablas, ya que la funcionalidad correspondiente a la creación de capas lo requiere.

5. Configuración de roles y usuarios en Tomcat: la seguridad se gestiona a nivel de contenedor. Es uno de los mecanismos más populares y menos costosos, ya que se aplica directamente el sistema veterano y conocido de securización de aplicaciones Web. Se deben establecer los roles y usuarios en el fichero de configuración de Tomcat 'tomcat-users.xml' situado en el directorio `${TOMCAT}\conf`.

```
<tomcat-users>
  <role rolename="profesor" />
  <role rolename="alumno" />
  <user username="profesor" password="profesor" roles="profesor,alumno" />
  <user username="alumno" password="alumno" roles="alumno" />
</tomcat-users>
```

*Figura 156. Configuración de roles y usuarios en Tomcat*

## 5.1.3 Ejecución de la aplicación

Una vez instalada la aplicación, es necesario verificar que se ha desplegado correctamente, bastará con acceder a la siguiente dirección *http://servidor:puerto/adgso* desde un navegador Web.

Una vez hecho, aparecerá la pantalla de validación de usuarios de la aplicación:

*Figura 157. Verificación de acceso a la aplicación ADGSO*

# 5.2 Manuales de Usuario

ADGSO está enfocado a dos roles de usuarios:

- Profesor: el usuario profesor tiene acceso a todas las funcionalidades de la aplicación, tanto a la creación de capas, como a la construcción y validación de geometrías.

- Alumno: el usuario alumno solo tiene acceso a las funcionalidades correspondientes a la manipulación de geometrías, construcción y validación de las mismas.

ADGSO está pensado para ayudar al usuario en el aprendizaje del almacenamiento de datos de tipo espacial en las bases de datos de Oracle, además de proporcionar una herramienta para crear escenarios de trabajo en dos dimensiones que sirvan de base de estudio para el análisis, manipulación y visualización de datos espaciales.

Como se verá más adelante en el apartado de aspectos generales, el usuario puede acceder al manual de usuario desde la aplicación. Según el rol que tenga se presentará el manual de usuario propio del profesor o del alumno. Dado que la única diferencia estriba en que el usuario alumno no tiene acceso a la funcionalidad de creación de capas, por simplificar, aunque realmente existan dos manuales distintos, en este apartado se va a presentar cada funcionalidad una única vez y se indica los permisos de acceso y ejecución según el tipo de usuario.

En primer lugar se va a presentar la autenticación y los aspectos generales.

## 5.2.1 Entrada al sistema

El acceso a ADGSO se realiza previa autenticación de usuario y contraseña de entrada al sistema.


*Figura 158. Acceso al sistema ADGSO*

Tras cumplimentar los datos, el usuario debe pulsar el botón '*Entrar*', si los datos de acceso son correctos, se accede a la pantalla principal de la aplicación, sino se muestra un mensaje informando de la invalidez del usuario y/o contraseña indicados.

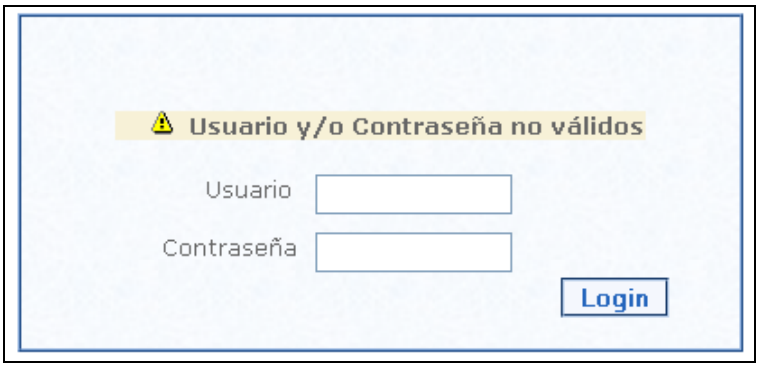



*Figura 159. Acceso incorrecto*

## 5.2.2 Estructura de las pantallas

Una vez cumplimentados los datos de acceso correctamente, se accede a la pantalla principal del sistema, en la que se presenta un breve resumen de las funcionalidades que proporciona ADGSO para el usuario autenticado.


*Figura 160. Pantalla principal de bienvenida al sistema*

Todas las pantallas presentan la misma estructura que la principal, una cabecera en la parte superior de la pantalla, un pie en la parte inferior, un menú principal de opciones a realizar en la parte central izquierda y por último, la zona de presentación y rellenado de datos que ocupa la mayor parte de la zona central.

## 5.2.3 Aspectos generales

### 5.2.3.1 Acceso a la ayuda

Todas las listas desplegables y áreas de texto de la aplicación que el usuario debe cumplimentar para llevar a cabo cualquiera de las funcionalidades del sistema, presentan un texto descriptivo que se muestra al situar el ratón encima del control. Para más información, se puede consultar la ayuda pulsando el enlace '*Ayuda*' o el icono que se muestra bajo la cabecera.

#### 5.2.3.2 Cambio de idioma

ADGSO está disponible en dos versiones, castellano e inglés. El idioma establecido por defecto es el castellano, pero puede cambiarse seleccionando el enlace o icono correspondiente al idioma que el usuario prefiera. El enlace '*Español/Spanish*' o icono para cambiar el idioma castellano ó el enlace '*Inglés/English*' o icono para el idioma inglés.

#### 5.2.3.3 Comunicación de mensajes

Tras llevar a cabo cualquiera de las funcionalidades disponibles en el sistema, se muestra un mensaje informativo ó mensaje de aviso en función de si la operación se ha realizado correctamente o se ha producido algún error. Los mensajes se muestran siempre por encima de los controles de rellenado y visualización de datos, en un área destacada en un tono amarillo pastel.

El icono avisa que el usuario debe subsanar algún error de cumplimentación de datos, mientras que el icono informa que la operación se ha realizado correctamente.

### 5.2.4 Creación de Capas

Es la única funcionalidad restringida a un único tipo de usuarios. Se accede seleccionando la opción 'Capas' del menú principal, que solo se presenta cuando el usuario que accede al sistema tiene el rol de profesor.

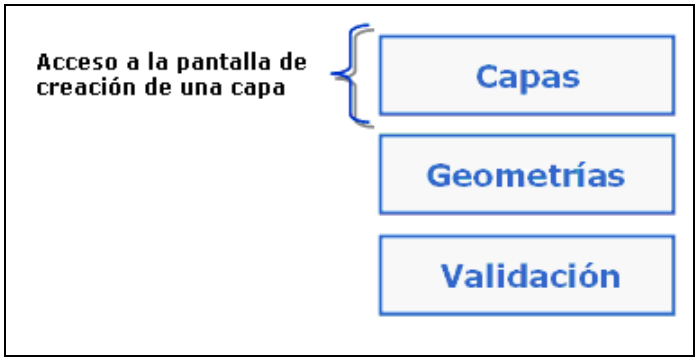



*Figura 161. Menú Creación de Capas*

Una vez pulsada la opción se accede a la pantalla de datos de la capa, en la que se solicita la cumplimentación de los datos necesarios para dar de alta una capa.

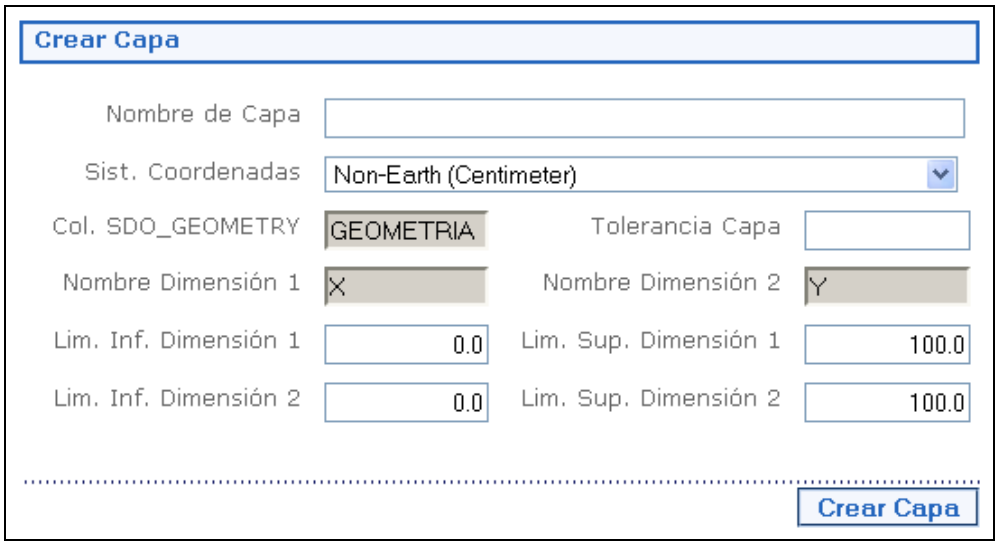



*Figura 162. Datos a cumplimentar para la creación de la capa*

Deben indicarse los siguientes datos:

- Nombre de la capa: nombre a asignar a la capa. Internamente es el nombre que se le da a la tabla que contiene la columna de tipo SDO_GEOMETRY en el que se van a insertar las geometrías.

- Metadatos de la capa: Para crear una capa, además de crear la tabla correspondiente a la capa, es necesario insertar sus medadatos:

    - Sistema de coordenadas: Sistema de referencia de la capa. Se debe seleccionar uno de los sistemas disponibles en Oracle Spatial. Dado que ADGSO da soporte a sistemas locales en dos dimensiones, en la lista desplegable solo figuran los correspondientes a sistemas locales.

    - Tolerancia Capa: Se debe indicar la tolerancia a establecer para las dimensiones de la capa. En sistemas locales se recomienda establecer el valor como la mitad de la diferencia menor entre dos valores cualesquiera de la dimensión.

    - Límite Inferior Dimensión 1: Límite inferior para los valores de las coordenadas de la primera dimensión. Se establece por defecto el valor ‘0’.

    - Límite Superior Dimensión 1: Límite superior para los valores de las coordenadas de la primera dimensión. Se establece por defecto el valor ‘100’.

    - Límite Inferior Dimensión 2: Límite inferior para los valores de las coordenadas de la segunda dimensión. Se establece por defecto el valor ‘0’.

    - Límite Superior Dimensión 2: Límite superior para los valores de las coordenadas de la segunda dimensión. Se establece por defecto el valor ‘100’.

El nombre de la columna de tipo SDO_GEOMETRY en el que se almacena la información espacial y los nombres de las dimensiones de la capa, se establecen a un valor fijo para todas las capas '*GEOMETRIA*', '*X*' e '*Y*' respectivamente, tal y como se puede observar en la figura "*Figura 162. Datos a cumplimentar para la creación de la capa*".

Tras cumplimentar todos los datos solicitados, se debe pulsar el botón '*Crear Capa*', Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto, se muestran los mensajes de avisos correspondientes detectados por sistema.

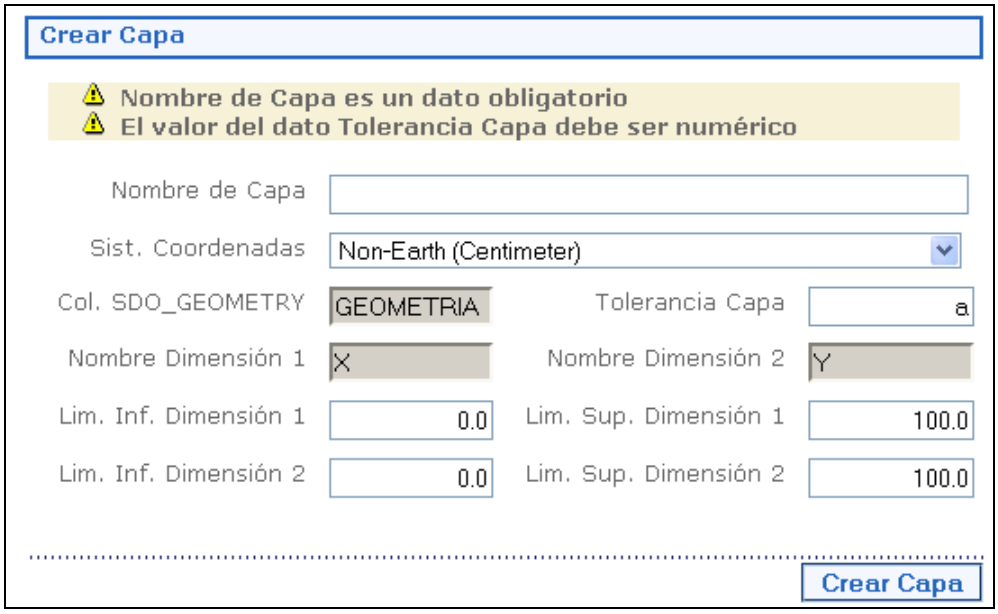



*Figura 163. Datos de la capa erróneos*

Una vez subsanados los errores que se muestran en los avisos, se podrá dar de alta la capa pulsando de nuevo el botón '*Crear Capa*'. Si todos los datos son correctos se da de alta la capa y se muestran las sentencias SQL que el usuario hubiera tenido que escribir y ejecutar manualmente si hubiese creado la capa desde fuera del sistema ADGSO.

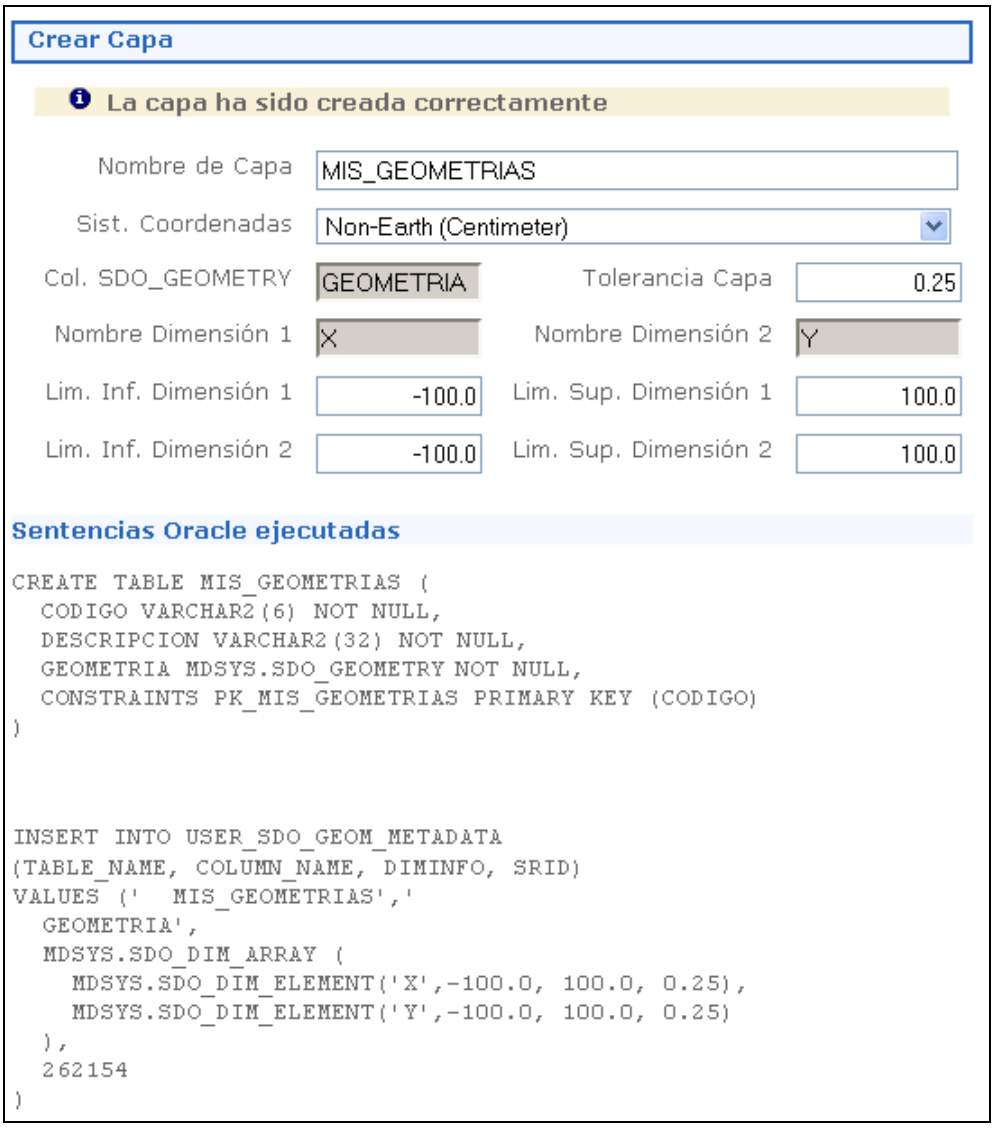



*Figura 164. Capa creada correctamente*

Como se puede observar en la figura anterior, en primer lugar se ejecuta la sentencia de creación de la capa, que tal y como se había mencionado anteriormente, corresponde a una tabla Oracle que presenta la columna '*GEOMETRIA*' en la que posteriormente se guardará la información espacial, es decir las geometrías. La tabla presenta además dos columnas '*CODIGO*' y '*DESCRIPCION*' que corresponden al identificador y la descripción corta de la geometría respectivamente, datos que deberán cumplimentarse al dar de alta las geometrías de la capa y que serán muy útiles si se utiliza el escenario creado desde ADGSO para continuar en el aprendizaje de Oracle Spatial.

Además de crear la tabla correspondiente a la capa, para realizar la carga de datos, validación, creación del índice espacial y operaciones de consulta sobre los datos de la capa, primero es necesario establecer sus metadatos. La segunda sentencia es la que introduce los metadatos en la vista *USER _SDO_GEOM_METADATA* proporcionada por Oracle.

A partir de este momento la capa queda disponible en el sistema para los usuarios con acceso a ADGSO puedan introducir geometrías en la misma.

## 5.2.5 Creación de Geometrías

Desde la opción ‘Geometrías’ del menú principal el usuario puede dar de alta la información espacial correspondiente a una capa, es decir las geometrías. Se presenta para ambos tipos de usuarios, tanto para profesores como para alumnos.

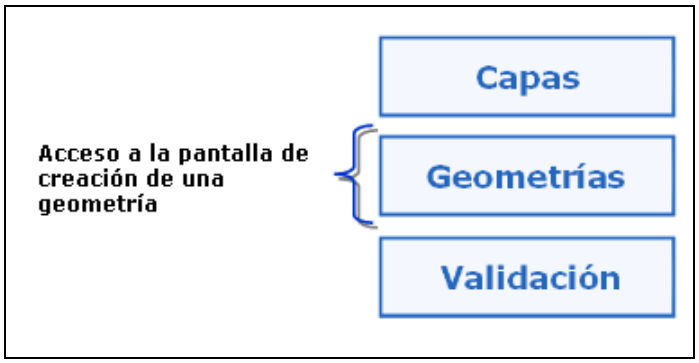



*Figura 165. Menú Creación de Geometrías*

En primer lugar se van a presentar los tipos de geometrías que permite crear ADGSO. El nombre utilizado para distinguirlas corresponde al utilizado en la documentación de referencia de Oracle Spatial. Dado que el sistema trabaja con escenarios en dos dimensiones son las siguientes:

- Puntos.
- Línea Simple, Polígono Simple: los trazos que unen los puntos que forman la geometría son todos rectos o todos curvos.
- Línea Compuesta, Polígono Compuesto: los trazos que unen los puntos que la forman son rectos y curvos, es decir se entremezclan.
- Rectángulo Optimizado: Spatial permite crear rectángulos, indicando sólo dos vértices, inferior izquierdo y superior derecho.
- Circunferencia Optimizada: De igual modo también permite crear circunferencias indicando solo tres vértices.
- Polígono con agujero: polígono que está formado por un anillo o polígono exterior y otro anillo a polígono interior.

### 5.2.5.1 Crear Geometría

Una vez que el usuario decida el tipo de geometría a insertar, tras seleccionar la opción de menú ‘Geometrías’, el primer paso consiste en seleccionar la capa en la que introducir la geometría.

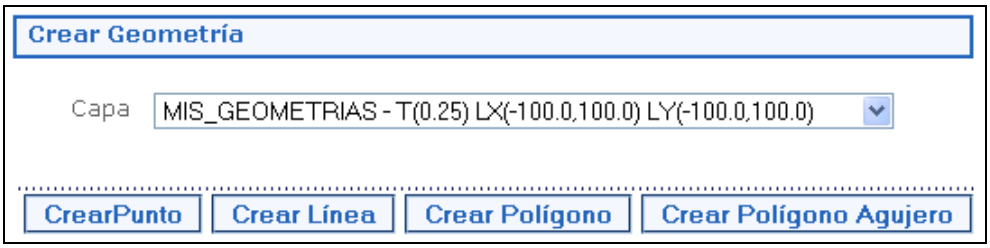



*Figura 166. Selección de Capa*

La pantalla presenta una lista desplegable con todas las capas disponibles en el sistema ADGSO. Además del nombre de la capa, se muestran los valores de la tolerancia y los límites inferior y superior de cada dimensión, valores que el usuario debe tener en cuenta al dar de alta las geometrías. Tras seleccionar la capa deseada se debe pulsar el botón correspondiente al tipo de dato espacial a insertar:

- '*Crear Punto*': si la geometría a insertar corresponde a un punto.
- '*Crear Línea*': si la geometría a insertar corresponde a una línea, tanto simple como compuesta.
- '*Crear Polígono*': si la geometría a insertar corresponde a un polígono simple, compuesto u optimizado.
- '*Crear Polígono Agujero*': si la geometría a insertar corresponde a un polígono con agujero.

En los siguientes apartados se verá la información específica a cumplimentar para cada tipo de geometría, pero hay dos datos comunes a indicar para todos ellos:

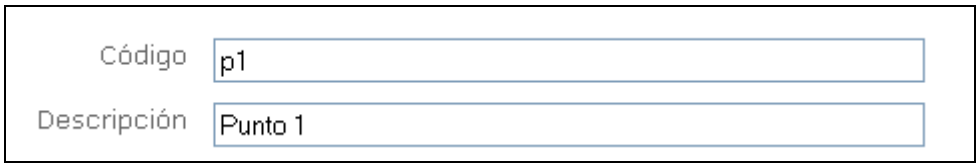



*Figura 167. Información no espacial de una geometría*

Estos datos corresponden a la información no espacial de la tabla:

- Código: como sucede con cualquier otro tipo de información que guarde en tablas de base de datos, cada registro de nuestra tabla debe poder identificarse mediante un código único.
- Descripción: además se ha añadido como dato obligatorio una descripción corta.

Serán muy útiles para utilizar el escenario que creado desde ADGSO para continuar en el aprendizaje de Oracle Spatial, por ejemplo para realizar análisis de datos.

A continuación se debe indicar la información espacial, es decir los datos que definen la geometría.

### 5.2.5.2 Crear Punto

Una vez seleccionada la capa y pulsado el botón '*Crear Punto*' se muestra la pantalla en la que cumplimentar los datos correspondientes al punto.

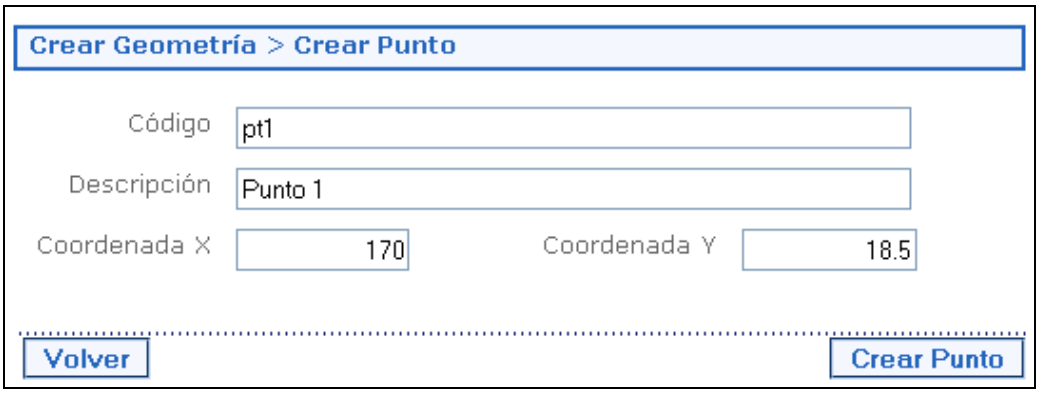



*Figura 168. Datos a cumplimentar para la creación de un punto*

Además de la información no espacial, 'Código' y 'Descripción' (véase apartado "*5.2.5.1 Crear Geometría*") se deben indicar las coordenadas 'X' e 'Y' del mismo. Tras cumplimentar todos los datos solicitados, se debe pulsar el botón 'Crear Punto'. Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto, se muestran los mensajes de avisos correspondientes detectados por sistema.

***NOTA IMPORTANTE**: A la hora de establecer los valores de las coordenadas, el usuario debe tener en cuenta los límites inferiores y superiores de las dimensiones X e Y de la capa.*

*Nota 1: Límites Inferiores y Superiores de las dimensiones*

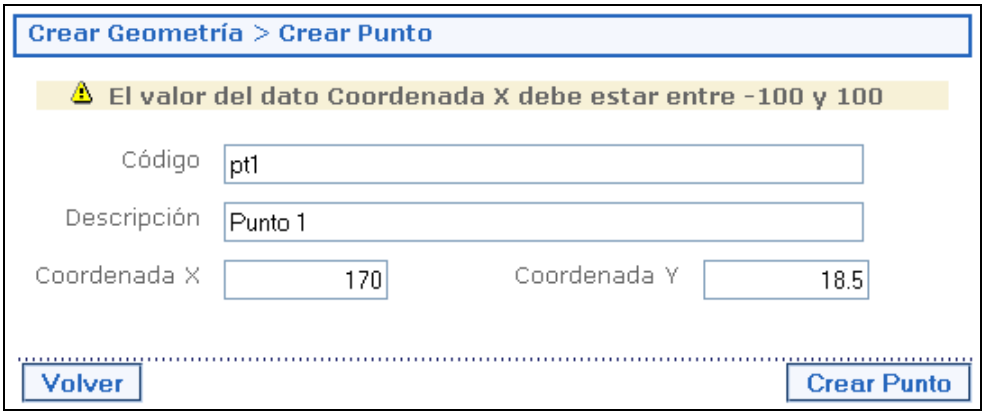



*Figura 169. Ejemplo error de datos en la creación de un punto*

El mensaje presentado indica que se ha superado el límite de la dimensión 'X'. Efectivamente, el valor '170' establecido en '*Coordenada X*' supera el valor '100' correspondiente al límite superior.

Una vez subsanados los errores, se inserta el punto en la capa y se muestran la sentencia SQL que el usuario hubiera tenido que escribir y ejecutar manualmente si lo hubiese insertado desde fuera del sistema ADGSO.

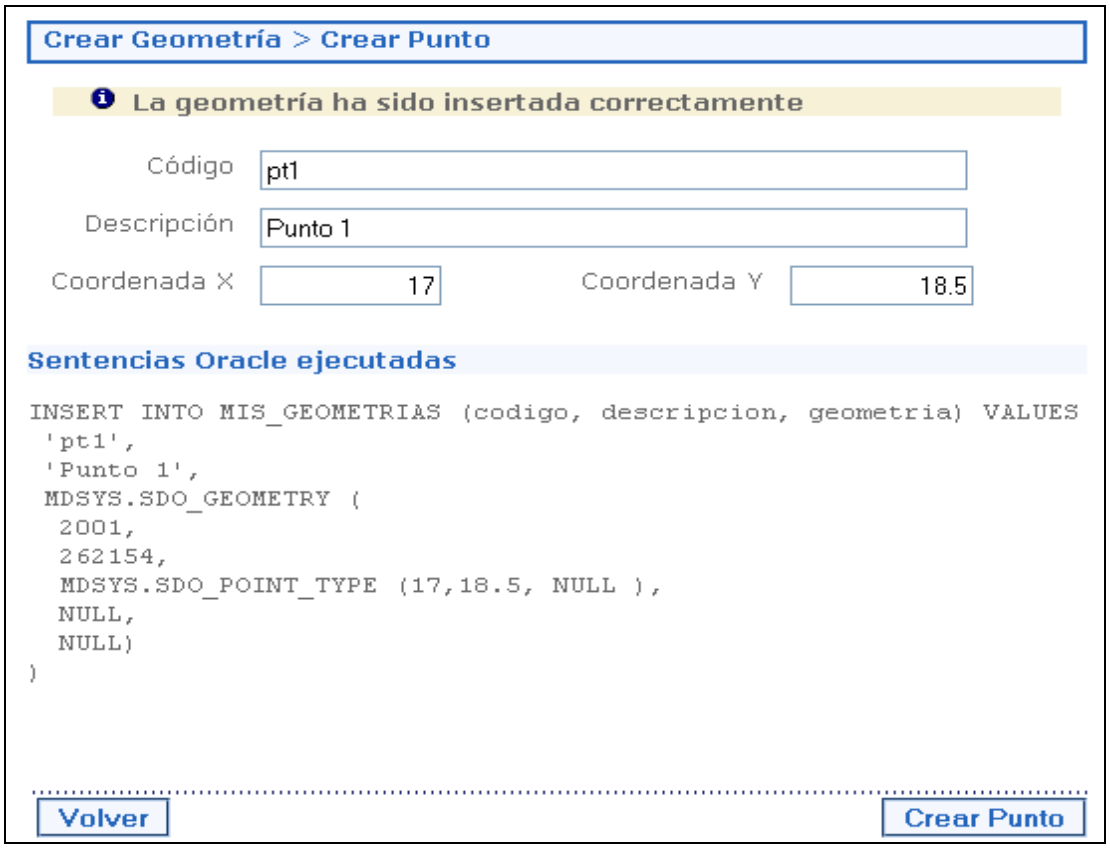



*Figura 170. Punto insertado correctamente*

La sentencia de inserción contiene el constructor *SDO_GEOMETRY* correspondiente al punto. Para crear puntos Oracle recomienda hacer uso del constructor *SDO_POINT_TYPE*.

### 5.2.5.3 Crear Línea

Una vez seleccionada la capa y pulsado el botón '*Crear Línea*' se muestra la pantalla en la que cumplimentar los datos correspondientes a la línea.

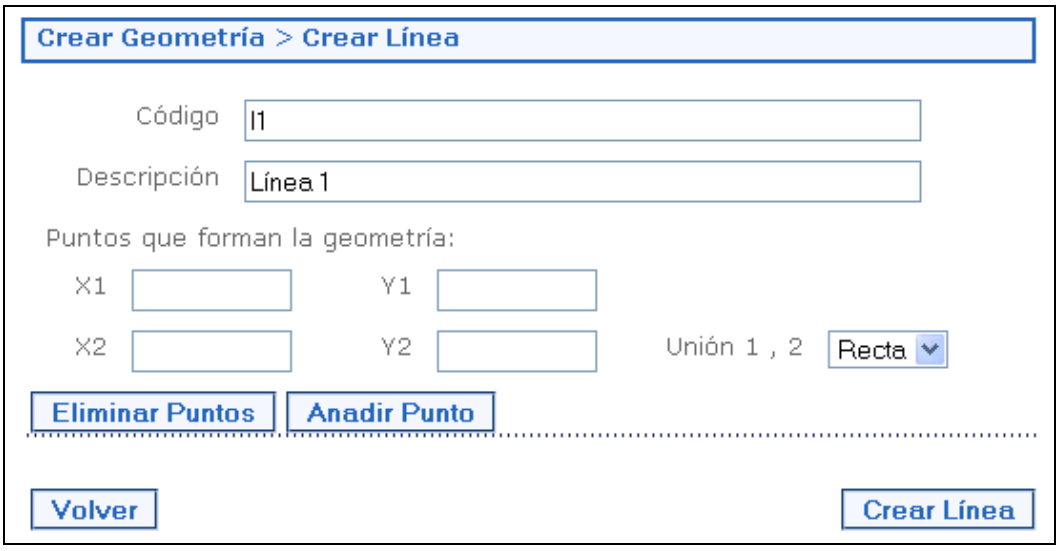



*Figura 171. Datos a cumplimentar para la creación de una línea*

Además de la información no espacial, '*Código*' y '*Descripción*' (véase apartado "*5.2.5.1 Crear Geometría*") se deben indicar los puntos o vértices que forman la línea y si el tipo de trazo que los une es recto o curvo. Una línea está formada como mínimo por dos puntos, por lo que al menos debe indicar los puntos 'X1, Y1' y 'X2, Y2' y el tipo de unión 'Unión 1,2'. Para añadir más puntos, el usuario debe pulsar el botón '*Añadir Punto*'.

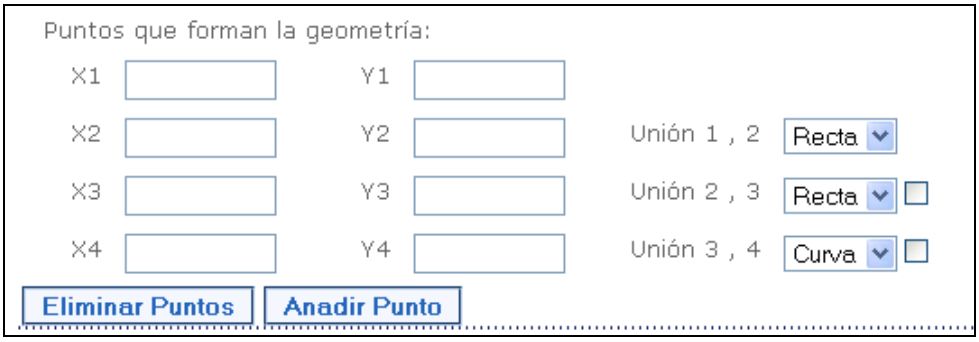



*Figura 172. Añadir punto a una línea*

No es necesario indicar los valores de las coordenadas de los puntos ya existentes en pantalla antes de añadir otro punto. No se comprueban los datos hasta pulsar el botón 'Crear Línea'. Si por error se añaden más puntos de los necesarios, pueden eliminarse marcando la caja de selección correspondiente a los puntos a eliminar y pulsando el botón '*Eliminar Puntos*', independientemente de que se hayan cumplimentado o no los datos.

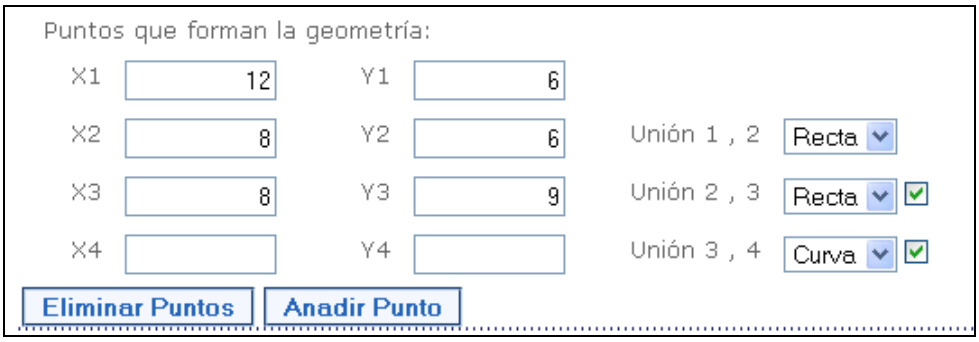



*Figura 173. Eliminar puntos de una línea*

Tras cumplimentar todos los datos solicitados, se debe pulsar el botón '*Crear Línea*'. Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto, se muestran los mensajes de avisos correspondientes detectados por sistema.

El usuario debe recordar que al igual que en el ejemplo de creación del punto, las coordenadas de los puntos que forman la línea deben estar dentro de los límites inferiores y superiores de Las dimensiones '*X*' e '*Y*' de la capa, véase la nota "*Nota 1: Límites Inferiores y Superiores de las dimensiones*".

Por último, antes de realizar la inserción de la geometría se realiza una validación semántica: Si la geometría no está formada correctamente (por ejemplo en el caso de las líneas, se han repetido vértices), se muestra el código Oracle del mensaje correspondiente al error. El usuario puede consultarlo en la referencia de Oracle para obtener más información y saber lo que debe hacer para solventar el error.

***NOTA IMPORTANTE**: El usuario debe recordar tener en cuenta el valor de la tolerancia de la capa a la hora de definir los vértices. Si la distancia entre un vértice y otro cualquiera de la geometría es menor que la tolerancia, se consideran vértices duplicados y por tanto la geometría no es válida.*

*Nota 2: Tolerancia de la capa*

Una vez subsanados los errores, se inserta la línea en la capa, se muestra la sentencia SQL que el usuario hubiera tenido que escribir y ejecutar manualmente si la hubiese insertado desde fuera del sistema ADGSO y acompañando a la misma, la tabla explicativa de los elementos que la forman.

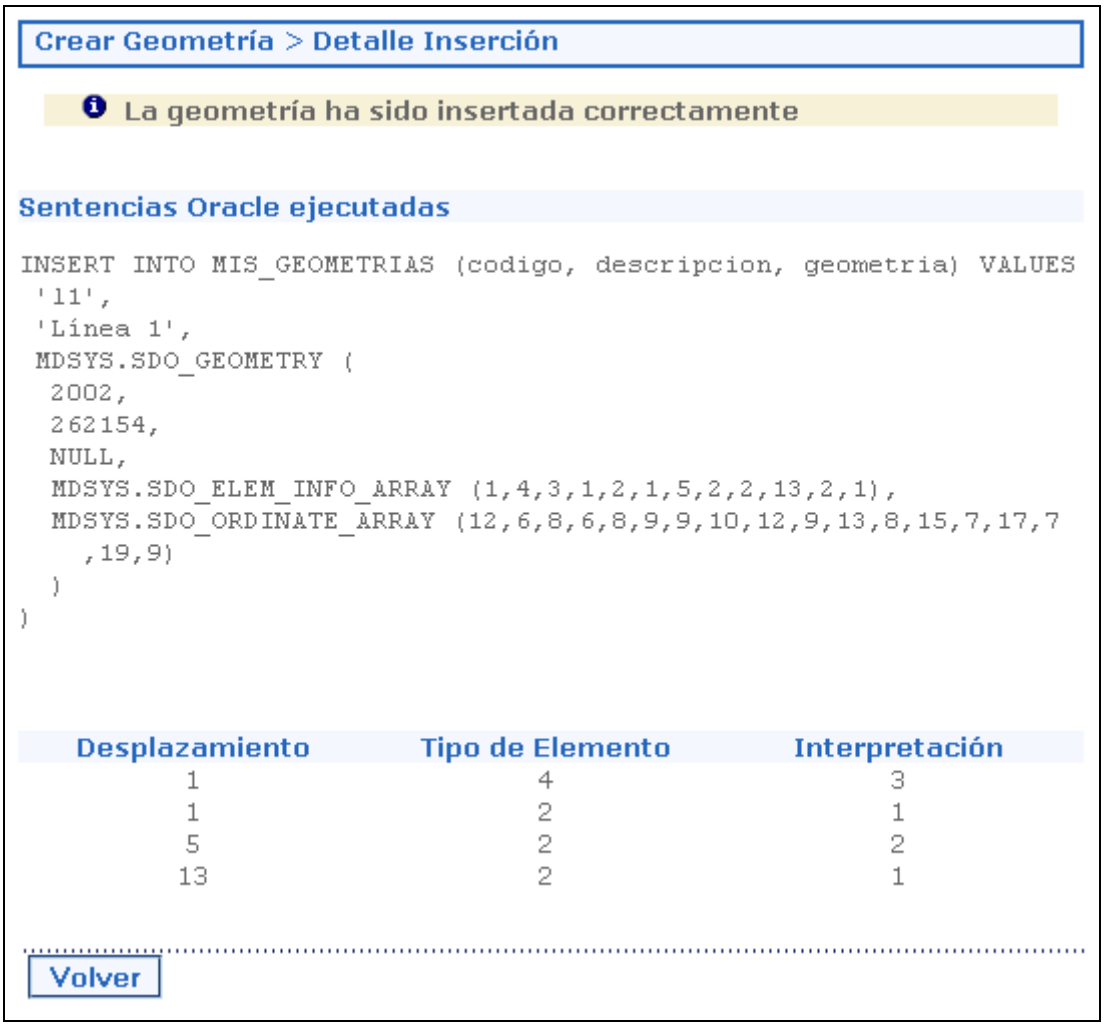



*Figura 174. Línea insertada correctamente*

La sentencia de inserción contiene el constructor *SDO_GEOMETRY* correspondiente a la línea. Para crear geometrías que no correspondan a un punto, es necesario hacer uso de los constructores *SDO_ELEM_INFO_ARRAY* y SDO_ORDINATE_ARRAY.

Como repaso, en este primer ejemplo de tabla explicativa, se va a dar una descripción detallada de lo que significa cada triplete de la tabla. Es un buen ejercicio que el usuario podría realizar al crear sus primeras geometrías, antes de escribir sus propios constructores.

```
SDO_ELEM_INFO_ARRAY
        1,      -- Línea de cabecera, por definición es 1
        4,      -- El tipo de elemento es 4 por ser una Línea compuesta
        3,      -- La interpretación es 3 porque es el número de cambios de tipos de
                -- subelementos que constituyen la línea compuesta. Este valor se especifica en la
                -- línea de cabecera de la geometría, a esta le siguen las tuplas de los elementos
                -- que componen la misma y en las que la interpretación toma los valores 1 y 2
                -- dependiendo de si el subelemento corresponde a un trazo recto o curvo
        1,      -- Las ordenadas del primer elemento que componen la geometría comienzan en la
                -- posición 1 del array
        2,      -- El tipo de elemento es 2 por ser una Línea
        1,      -- La interpretación es 1 porque sus vértices están conectados por trazos rectos
        5,      -- Las ordenadas del segundo elemento que componen la geometría comienzan
                -- en la posición 5 del array
        2,      -- El tipo de elemento es 2 por ser una Línea
        2,      -- La interpretación es 2 porque sus vértices están conectados por arcos
        13,     -- La ordenadas del tercer elemento que componen la geometría comienzan en la
                -- posición 13 del array
        2,      -- El tipo de elemento es 2 por ser una Línea
        1       -- La interpretación es 1 porque sus vértices están conectados por trazos rectos

SDO_ORDINATES_ARRAY
        12 – 6  -- Punto 1: Posiciones 1 y 2 del array (Inicio primer subelemento)
        8 – 6   -- Punto 2: Posiciones 3 y 4 del array
        8 – 9   -- Punto 3: Posiciones 5 y 6 del array.
                -- Fin primer subelemento, Inicio segundo subelemento
        9 – 10  -- Punto 4: Posiciones 7 y 8 del array
        12 – 9  -- Punto 5: Posiciones 9 y 10 del array
        13 – 8  -- Punto 6: Posiciones 11 y 12 del array
        15 – 7  -- Punto 7: Posiciones 13 y 14 del array.
                -- Fin segundo subelemento, Inicio tercer subelemento
        17 - 7  -- Punto 8: Posiciones 15 y 16 del array
        19 – 9  -- Punto 9: Posiciones 17 y 18 del array. Fin tercer subelemento
```

*Figura 175. Ejemplo de descripción detallada de la tabla explicativa*

### 5.2.5.4 Crear Polígono

Una vez seleccionada la capa y pulsado el botón '*Crear Polígono*' se muestra la pantalla en la que cumplimentar los datos correspondientes al polígono.

Además de la información no espacial, '*Código*' y '*Descripción*' (véase apartado "*5.2.5.1 Crear Geometría*") se debe indicar el tipo de polígono que quiere introducir y los puntos o vértices que definen la geometría.

Se puede elegir entre tres tipos:

- Polígono: para crear un polígono simple o compuesto no optimizado. Un polígono simple o compuesto está formado por n puntos, Se deben indicar las coordenadas de los puntos que lo forman '*X1, Y1*', '*X2, Y2*', '*Xn, Yn*' y el tipo de unión entre ellos '*Unión 1,2*', '*Unión n-1,n*'. El uso de los botones '*Añadir Punto*' y '*Eliminar Puntos*' es el mismo que para la creación de líneas, véanse las figuras "*Figura 172. Añadir punto a una línea*" y "*Figura 173. Eliminar puntos de una línea*".

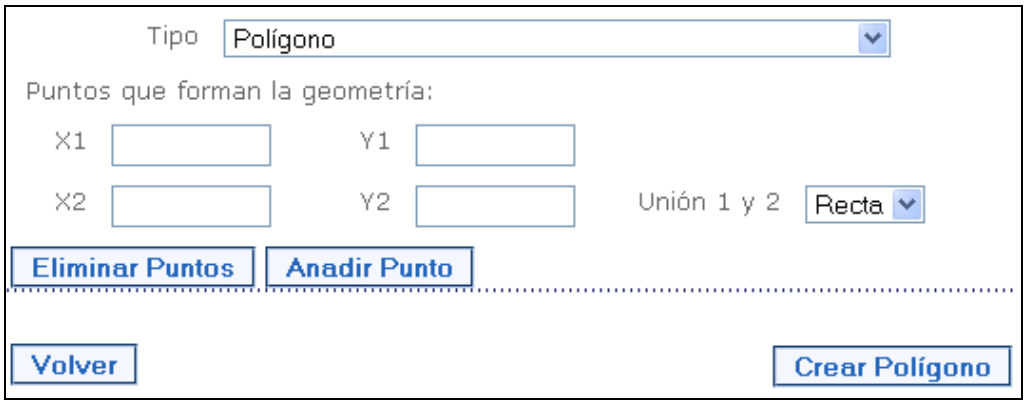



*Figura 176. Datos a cumplimentar para la creación de un polígono*

- Rectángulo optimizado: Spatial permite crear rectángulos, indicando sólo dos vértices, inferior izquierdo y superior derecho. Únicamente se deben indicar los puntos 'X1, Y1' y 'X2, Y2'.

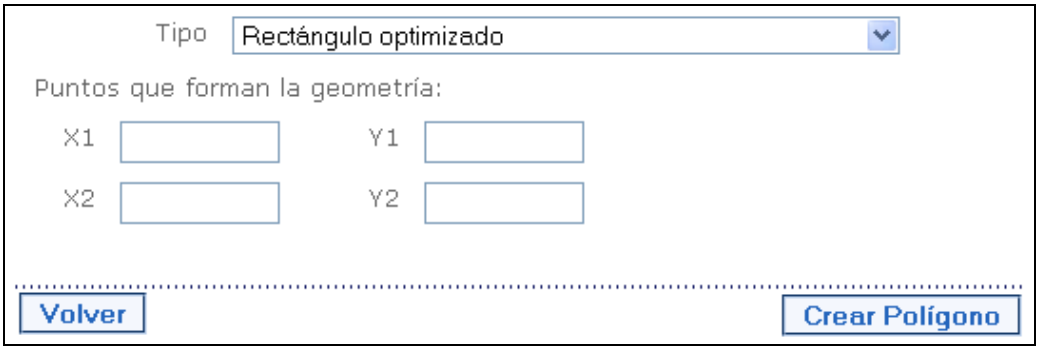



*Figura 177. Datos a cumplimentar para la creación de un rectángulo optimizado*

- Circunferencia optimizada: Spatial también permite crear circunferencias, indicando sólo tres vértices. Únicamente debe indicar los puntos '*X1, Y1*', '*X2, Y2*' y '*X3, Y3*'.

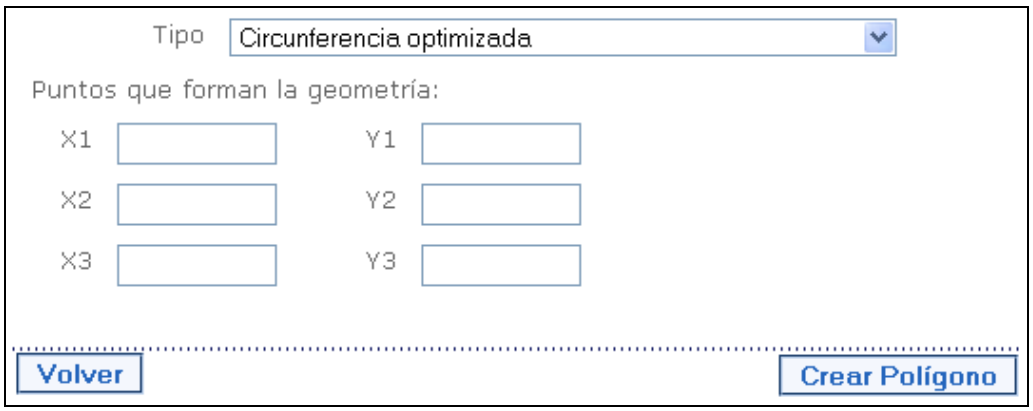



*Figura 178. Datos a cumplimentar para la creación de una circunferencia optimizada*

Tras cumplimentar todos los datos solicitados, se debe pulsar el botón '*Crear Polígono*'. Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto, se muestran los mensajes de avisos correspondientes detectados por sistema.

Al igual que en los ejemplos anteriores de creación del punto y la línea, se deben tener en cuenta los límites inferiores y superiores de las dimensiones '*X*' e '*Y*' y la

tolerancia de la capa, véanse las notas "*Nota 1: Límites Inferiores y Superiores de las dimensiones*" *y* "*Nota 2: Tolerancia de la capa*" respectivamente.

Además al igual también que en los ejemplos anteriores, antes de realizar la inserción de la geometría se realiza una validación semántica: Si la geometría no está formada correctamente se muestra el código Oracle del mensaje correspondiente al error.

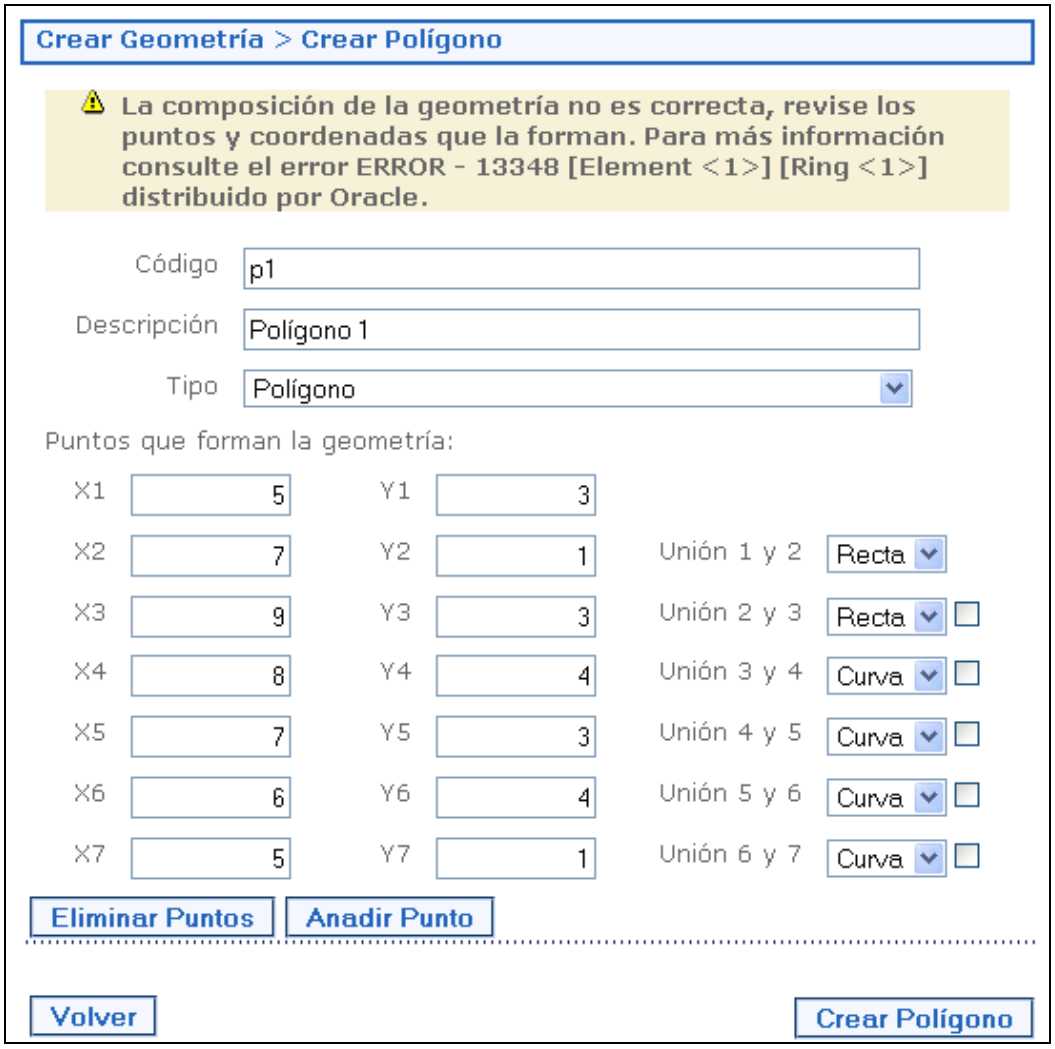



*Figura 179. Error semántico al insertar el polígono*

Al consultar la referencia de Oracle el mensaje indica que la geometría no es correcta, dado que el polígono no está cerrado correctamente, es decir, en primer y último punto de la misma no coinciden.

Una vez subsanados los errores, se inserta el polígono en la capa, se muestra la sentencia SQL que el usuario hubiera tenido que escribir y ejecutar manualmente si lo hubiese insertado desde fuera del sistema ADGSO y acompañando a la misma, la tabla explicativa de los elementos que lo forman.

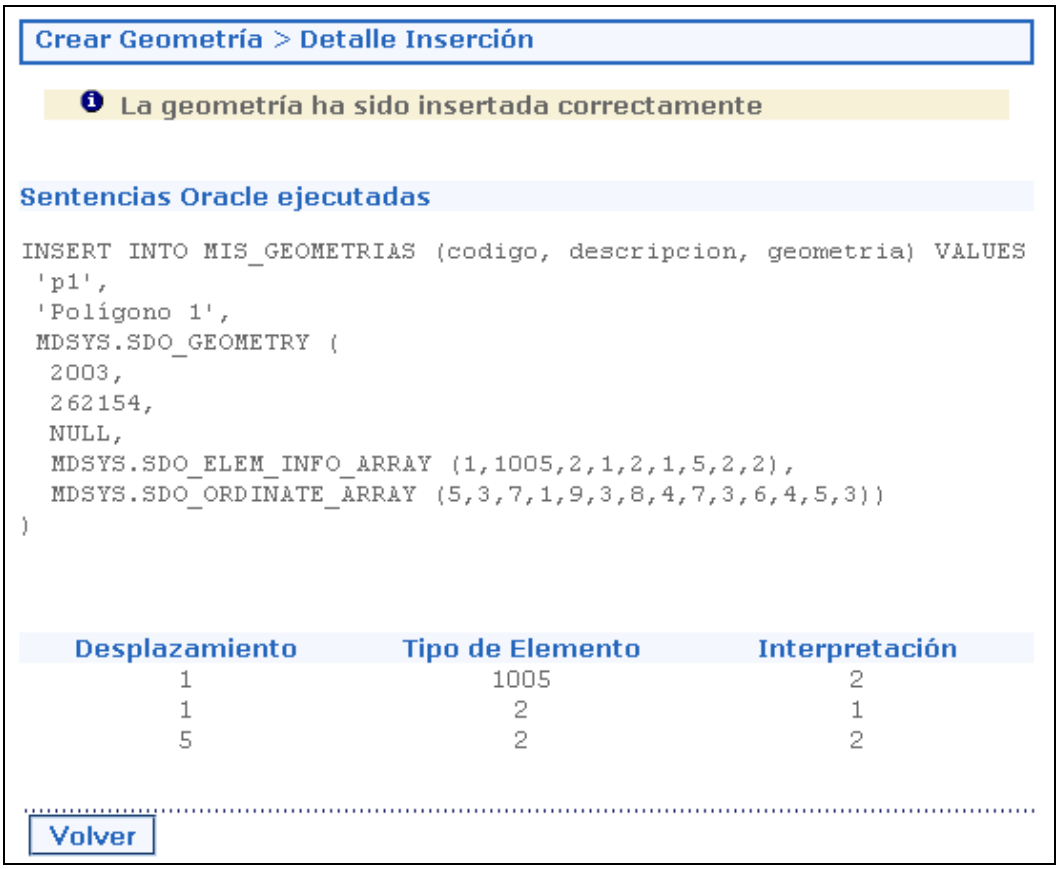



*Figura 180. Polígono insertado correctamente*

Para recordar el significado de los elementos de la tabla explicativa, véanse las figuras "*Figura 174. Línea insertada correctamente*" y "*Figura 175. Ejemplo de descripción detallada de la tabla explicativa*".

### 5.2.5.5 Crear Polígono con Agujero

Una vez seleccionada la capa y pulsado el botón '*Crear Polígono Agujero*' se muestra la pantalla en la que cumplimentar los datos correspondientes al polígono.

Un polígono con agujero está formado por el anillo externo (polígono exterior) y el anillo interno (polígono interior).

En primer lugar se muestra la pantalla para cumplimentar los datos del polígono exterior.

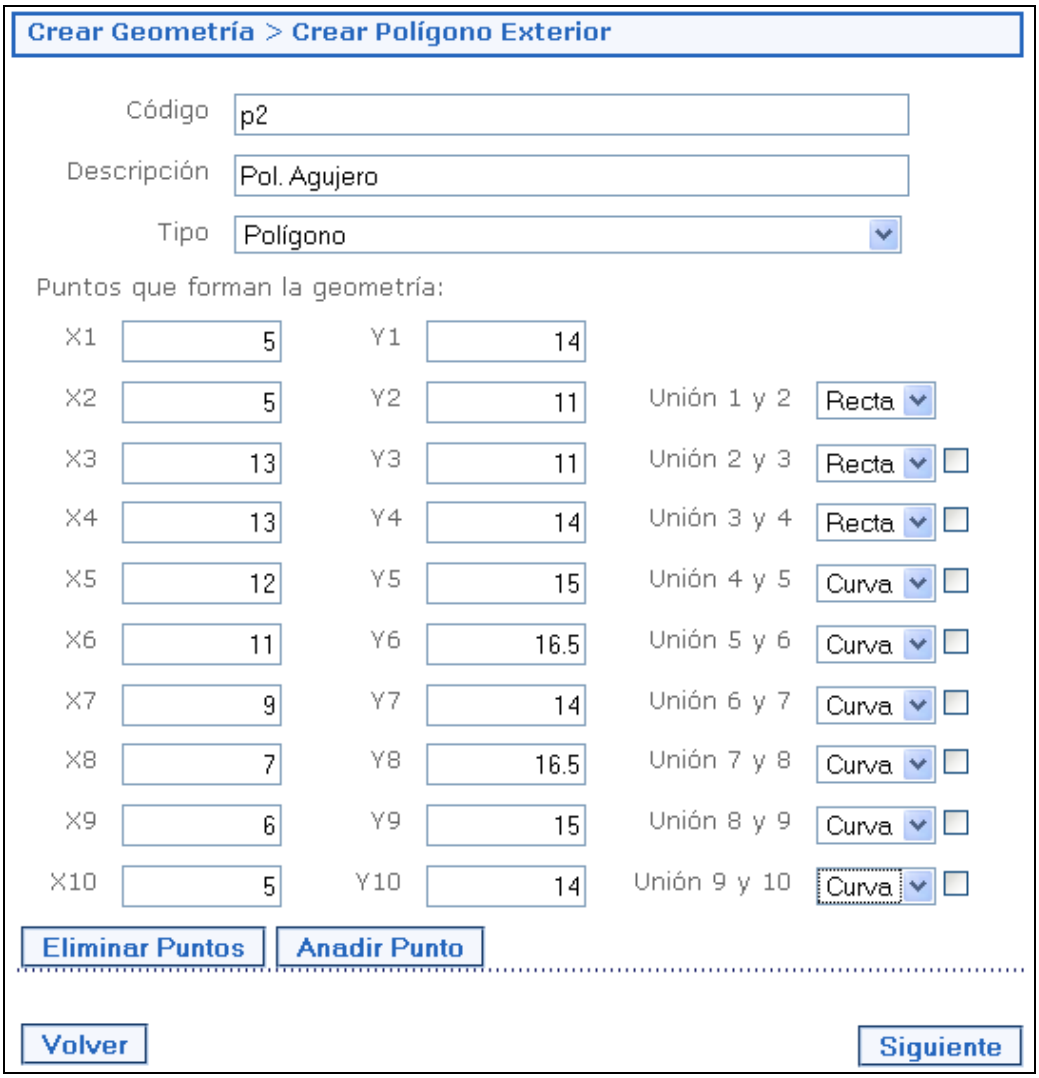



*Figura 181. Polígono externo*

Los datos a cumplimentar son los mismos que los indicados en la creación de un polígono, véase el apartado "*5.2.5.4 Crear Polígono*".

Tras cumplimentar la información del polígono externo, se debe pulsar el botón '*Siguiente*' para acceder a la pantalla de datos del polígono interno.

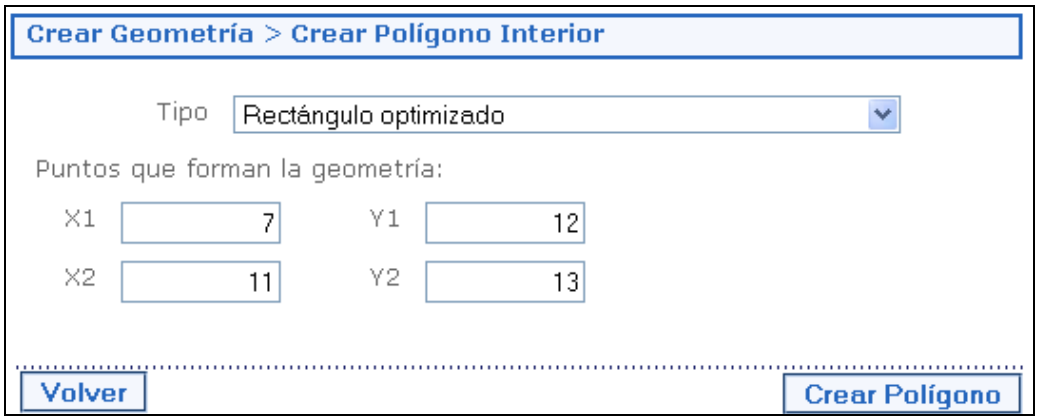



*Figura 182. Polígono interno*

Los datos a cumplimentar son los mismos que los indicados en la creación de un polígono, véase el apartado "*5.2.5.4 Crear Polígono*".

Tras cumplimentar todos los datos solicitados, se debe pulsar el botón '*Crear Polígono*'. Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto, se muestran los mensajes de avisos correspondientes detectados por sistema.

Al igual que en los ejemplos anteriores de creación del punto, la línea y el polígono, el usuario debe tener en cuenta los límites inferiores y superiores de las dimensiones '*X*' e '*Y*' y la tolerancia de la capa, véanse las notas "*Nota 1: Límites Inferiores y Superiores de las dimensiones*" *y* "*Nota 2: Tolerancia de la capa*" respectivamente.

Además al igual también que en los ejemplos anteriores, antes de realizar la inserción de la geometría se realiza una validación semántica: Si la geometría no está formada correctamente se muestra el código Oracle del mensaje correspondiente al error.

Una vez subsanados los errores, se inserta el polígono en la capa, se muestra la sentencia SQL que el usuario hubiera tenido que escribir y ejecutar manualmente si lo hubiese insertado desde fuera del sistema ADGSO y acompañando a la misma, la tabla explicativa de los elementos que lo forman.

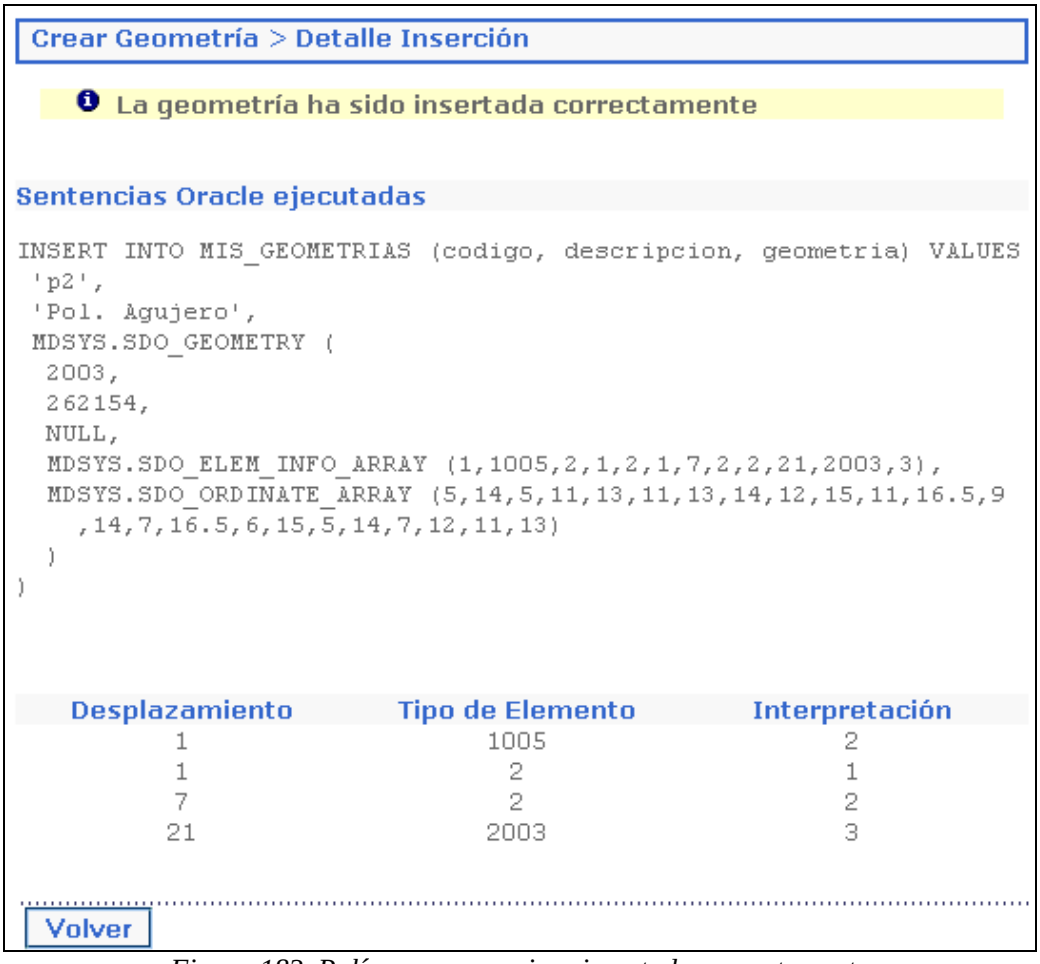



*Figura 183. Polígono con agujero insertado correctamente*

## 5.2.6 Validación de Constructores

Una vez que el usuario haya aprendido el uso del constructor SDO_GEOMETRY, desde la opción 'Validación' del menú principal podrá escribir sus propios constructores y validar si son correctos. Se presenta para ambos tipos de usuarios, tanto para profesores como para alumnos.

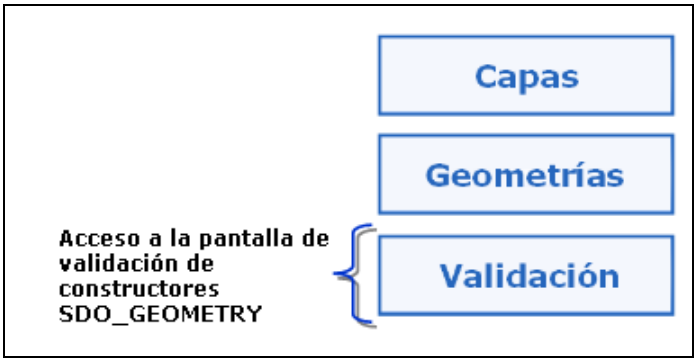



*Figura 184. Menú Validación Constructores*

El usuario accede a la pantalla de datos de validación, en la que se solicita que cumplimente los datos necesarios para validar el constructor.

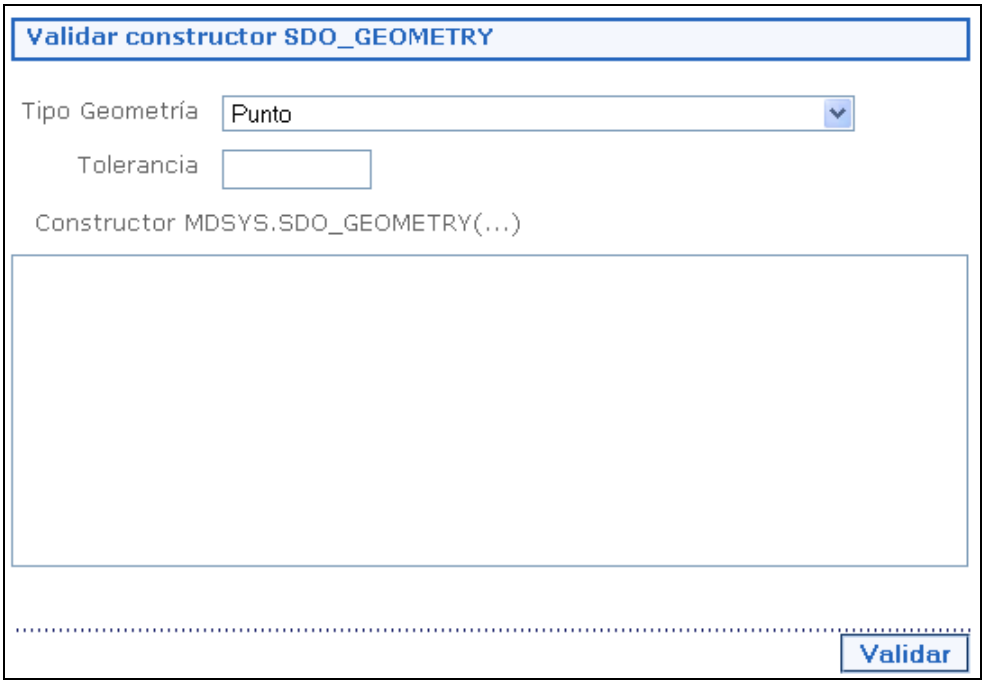



*Figura 185. Menú Validación Constructores*

Se debe indicar:

- Tipo Geometría: el tipo de geometría al que corresponde el constructor a validar. Para más detalle véase la clasificación de geometrías en el apartado "*5.2.5 Creación de Geometrías*".

- Tolerancia: valor de la tolerancia a aplicar en la validación de la geometría. Para recordar la importancia que tiene establecer este valor correctamente, véase la nota "*Nota 2: Tolerancia de la capa*".

- Constructor SDO_GEOMETRY: constructor SDO_GEOMETRY a validar.

Tras cumplimentar todos los datos solicitados, se debe pulsar el botón '*Validar*'. Si no se ha dado valor a todos los datos o se ha indicado un valor incorrecto en el valor de la tolerancia, se mostrarán los mensajes de avisos correspondientes detectados por sistema.

Una vez subsanados los errores de cumplimentación de datos, se procede a la validación del constructor.

En primer lugar se realiza la validación sintáctica, es decir, se comprueba si la sentencia del constructor está bien formada para el tipo de geometría indicado y si están definidos correctamente todos los elementos del constructor.

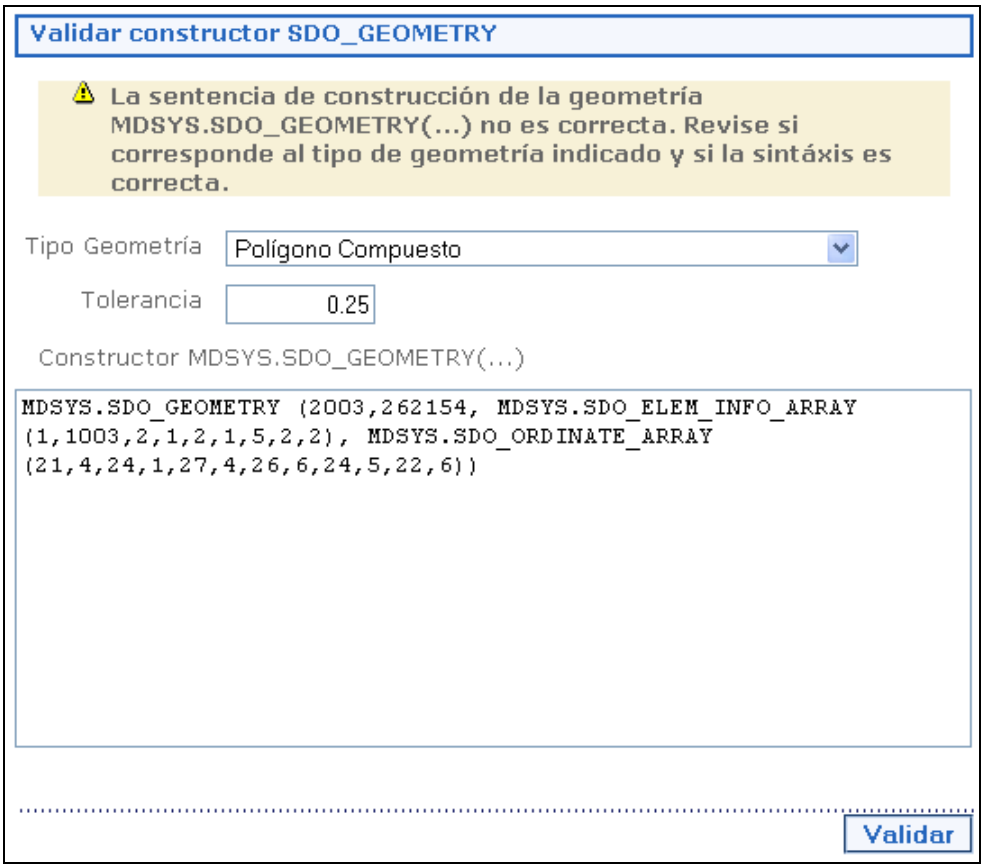



*Figura 186. Error sintáctico*

El constructor de la figura anterior no está bien formado, el sistema detecta un error sintáctico porque falta el campo correspondiente al constructor SDO_POINT. En este caso como la geometría no es un punto, debe indicarse a NULL.

```
MDSYS.SDO_GEOMETRY (2003,262154, NULL, MDSYS.SDO_ELEM_INFO_ARRAY
(1,1003,2,1,2,1,5,2,2), MDSYS.SDO_ORDINATE_ARRAY
(21,4,24,1,27,4,26,6,24,5,22,6))
```

Además de la validación sintáctica, ADGSO realiza una primera validación semántica en la que comprueba si el constructor corresponde realmente al tipo de geometría indicado. Para ello revisa los valores de los atributos *GTYPE* y *SDO_ELEM_INFO_ARRAY* (en el caso de que la geometría no sea un punto).

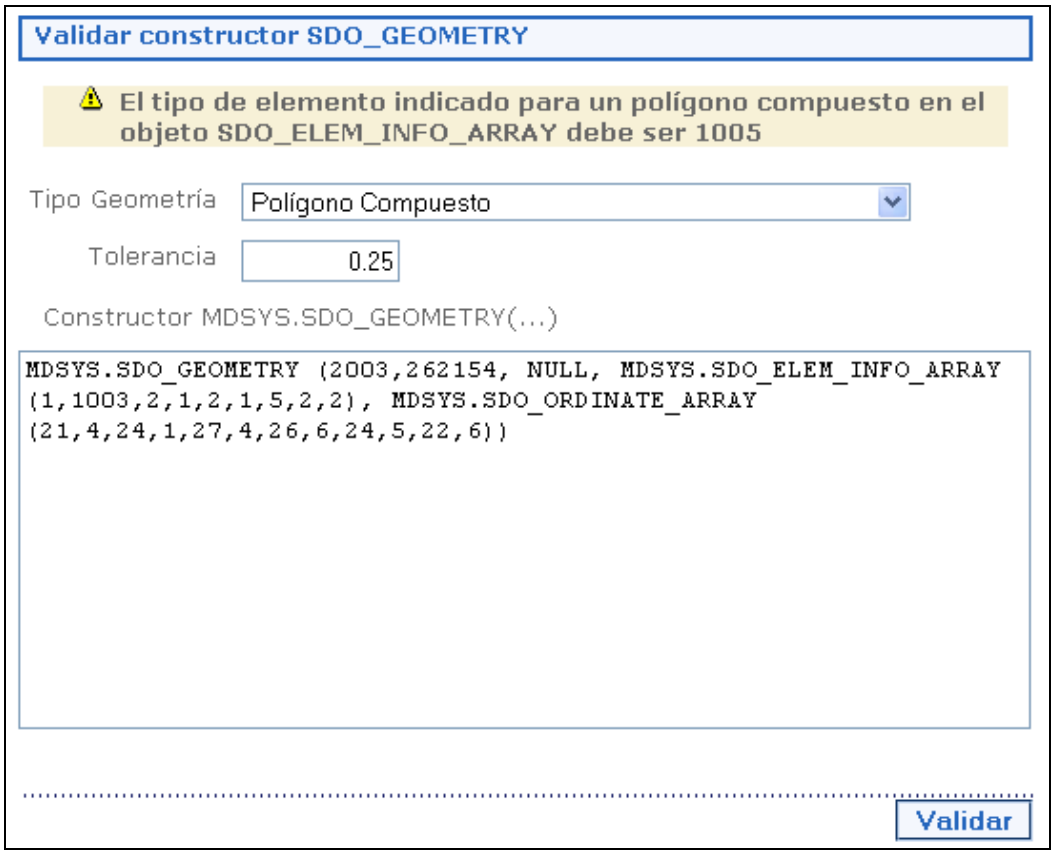



*Figura 187. Error semántico en el constructor SDO_ELEM_INFO_ARRAY*

En este caso el constructor está bien formado, pero tal y como muestra el aviso de la figura anterior, el tipo de elemento indicado en la línea de cabecera del constructor *SDO_ELEM_INFO_ARRAY* no corresponde al que debería indicarse para un polígono compuesto.

```
MDSYS.SDO_GEOMETRY (2003,262154, NULL, MDSYS.SDO_ELEM_INFO_ARRAY
(1,1003,2,1,2,1,5,2,2), MDSYS.SDO_ORDINATE_ARRAY
(21,4,24,1,27,4,26,6,24,5,22,6))
```

Por último se realiza una segunda validación semántica, en la que internamente ADGSO hace uso de la función de validación proporcionada por Oracle para validar un constructor SDO_GEOMETRY. Esta función comprueba cosas tales como que un polígono esté cerrado correctamente, que no haya repetición de vértices, que las líneas del polígono no se crucen entre sí, etc.

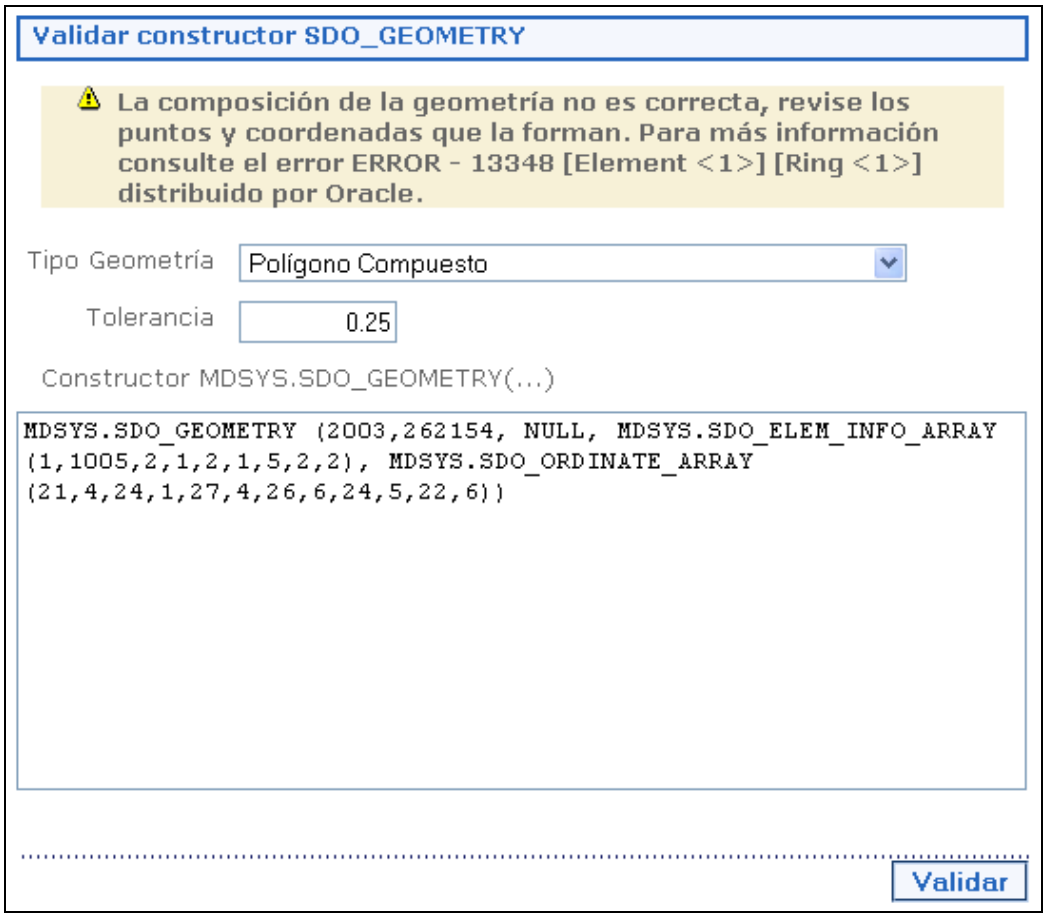



*Figura 188. Error semántico en el constructor SDO_GEOMETRY*

El constructor de la figura anterior está bien formado y corresponde al tipo de geometría seleccionado, pero no es correcto porque el polígono no se ha cerrado, es decir, el primer vértice del polígono no coincide con el último. Es la explicación que se da en la referencia de Oracle para el error '13348'.

El sistema ADGSO seguirá mostrando avisos hasta que el constructor sea correcto tanto sintáctica como semánticamente. Una vez subsanados, cuando el constructor sea correcto, se muestra una tabla explicativa de los elementos que componen la geometría.

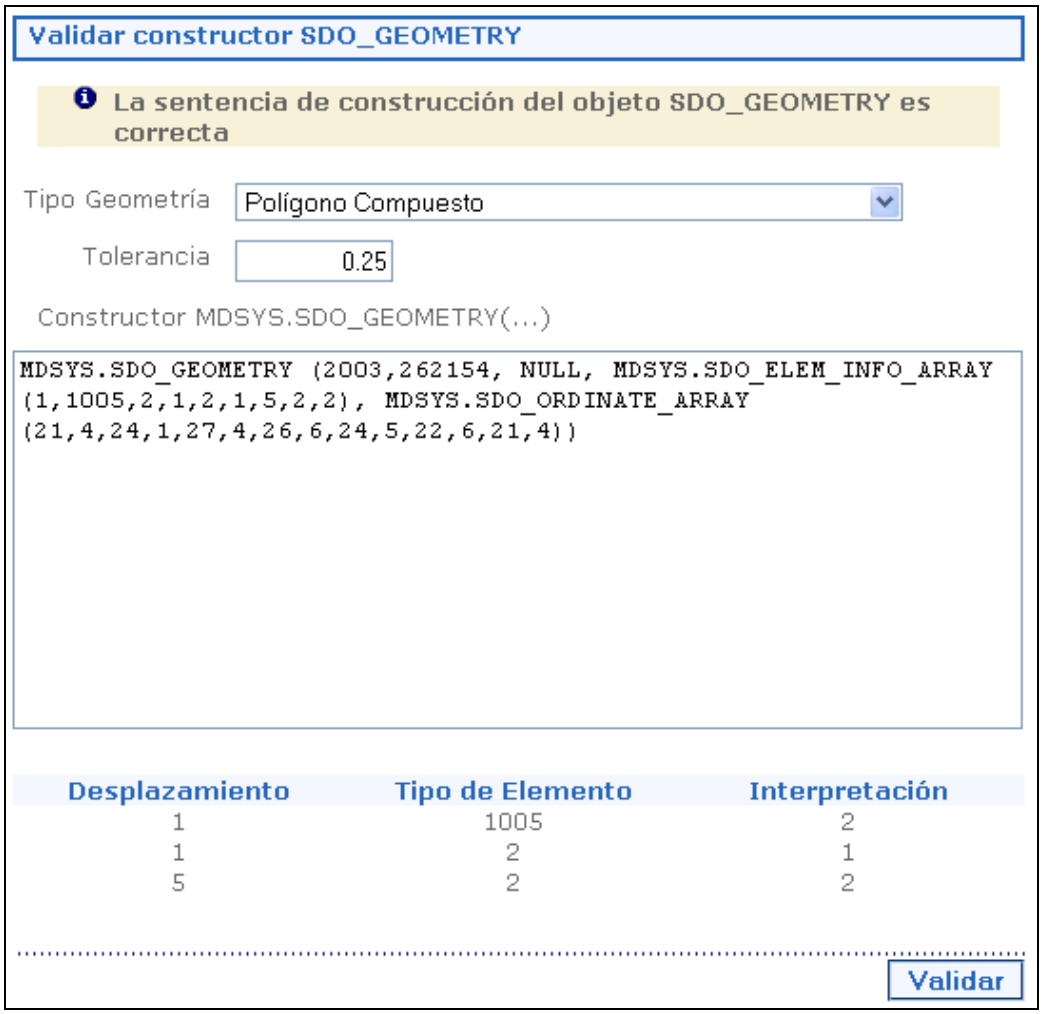



*Figura 189. Constructor SDO_GEOMETRY correcto*

Para recordar el significado de los elementos de la tabla explicativa, véanse las figuras "*Figura 174. Línea insertada correctamente*" y "*Figura 1. Componentes de un SIG*".

# Capítulo 6

# Conclusiones

El objetivo que se pretendía alcanzar con la elaboración de este proyecto, era llevar a cabo un estudio sobre cómo se gestiona la información espacial en un sistema gestor de bases de datos, concretamente en Oracle Spatial; y proporcionar una herramienta didáctica que ayudara y asistiera en el aprendizaje del almacenamiento de datos de tipo espacial, además de proporcionar una forma sencilla de crear escenarios de trabajo que sirvieran de base de estudio para el análisis, manipulación y visualización de datos espaciales.

Llevar a cabo el objetivo marcado, ha requerido un proceso de investigación sobre la gestión de la información espacial dividido en dos partes.

En primer lugar se ha observado la multitud de ámbitos en los que tiene aplicación la información espacial, así como la gran cantidad de software disponible en el mercado para explotarla, tanto para resolver problemas complejos y delicados, como para cuestiones más cotidianas.

Lo más importante a destacar en esta parte, es la tendencia de las organizaciones dedicadas al desarrollo de los SIG, a pasar de una colección de archivos tradicionales basados en los modelos vector, raster, y datos de diseño asistido por ordenador (CAD) a un entorno integrado, donde todos los datos espaciales y de negocios se gestionan como una base de datos continua. Una vez que los datos espaciales se almacenan en una base de datos, puede tratarse, recuperarse y relacionarse como el resto de datos, por lo que la información SIG deja de ser un privilegio de los expertos y pueden ser explotados por usuarios y aplicaciones de toda la organización.

En segundo lugar, una vez entrado en materia, se ha profundizado en la solución aportada por Oracle.

En el apartado dedicado a Oracle Spatial, se ha dado una visión detallada del producto, cubriendo los aspectos más básicos e importantes, haciendo especial incapié en la ilustración de ejemplos de distintos tipos de geometrías, en base a facilitar el aprendizaje y comprensión de los diferentes elementos que componen el objeto SDO_GEOMETRY y el significado de cada uno de ellos dentro de las sentencias. Observando dichos ejemplos se ha podido comprobar que efectivamente la tarea de generar un script para cargar un escenario de datos puede ser una tarea bastante ardua. Lo que ha llevado a la implementación de ADGSO como culminación de la consecución del objetivo del proyecto.

Llegados a este punto, se puede decir que el objetivo del cual partíamos al inicio de este documento ha sido alcanzado en su totalidad.

Para finalizar, ADGSO se centra en la carga y validación de datos espaciales, podría ser interesante en un futuro, añadir a las funcionalidades ya existentes, otras enfocadas al análisis y visualización de datos:

- Creación de índices: Como se ha mencionado anteriormente ADGSO además de tener una función didáctica, proporciona una herramienta para crear escenarios de trabajo con los que se pueda trabajar para continuar con el aprendizaje del análisis de datos espaciales. Para trabajar con operadores espaciales es necesario que las tablas tengan índices espaciales creados en las tablas correspondientes a las capas que se crean desde ADGSO. Se podría añadir una nueva funcionalidad que permitiera crear un índice espacial sobre una capa, de modo que el usuario de la aplicación no tuviera que crearlo desde una herramienta externa a ADGSO.

- Análisis espacial: una vez creados los índices, se pueden realizar consultas sobre el escenario de trabajo. Se podría añadir una nueva funcionalidad que permitiera introducir consultas a ejecutar, de modo que el usuario de la aplicación no tuviera que ejecutarlas desde una herramienta externa a ADGSO.

- Visualización de datos: Al crear una geometría desde ADGSO, además de presentar la sentencia necesaria para crearla, se muestra una tabla explicativa de los elementos que componen la geometría. Acompañando a la tabla se podría incluir la imagen de la geometría, de forma que la explicación fuese todavía más clara. Además se podría incluir una funcionalidad que mostrase todas las geometrías de una capa, es decir, el escenario completo.

# Capítulo 7

# Presupuesto

En este capítulo se listan los costes asociados a los medios materiales y a los recursos humanos utilizados en el desarrollo de este proyecto.

Se ha planificado un calendario laboral que implica una jornada de cinco horas diarias. Aunque la línea de realización del proyecto ha sido irregular a lo largo de todo el proceso, se ha supuesto una dedicación continua. Por lo que la duración del proyecto y el número de jornadas refleja la media real de las horas invertidas en el desarrollo del proyecto al completo.

La distribución de los trabajos realizados se ha divido en las siguientes fases.

- Fase de análisis: esta fase incluye además del análisis del aplicativo ADGSO, toda la investigación previa realizada sobre las bases de datos espaciales y Oracle Spatial, así como la generación de la documentación tanto de la memoria como del aplicativo ADGSO. El computo de 70 días * 5horas/día hacen un total de 350 horas.

- Fase de diseño técnico: esta fase incluye la parte concerniente al diseño del aplicativo ADGSO. El computo de 15 días * 5horas/día hacen un total de 75 horas.

- Fase de implementación y pruebas: esta fase incluye la implementación del sistema ADGSO, la definición de la batería de pruebas a realizar, la instalación del aplicativo y la realización de las pruebas. El computo de 25 días * 5horas/día hacen un total de 125 horas.

El desglose presupuestario de costes de materiales y recursos materiales utilizados se presenta a continuación en la siguiente hoja de cálculo. Teniendo en cuenta la planificación realizada en jornadas de cinco horas diarias, el valor de un ‘hombre/mes’ se establece a cien horas de trabajo, habiéndose hecho los cálculos en base a este dato.

En los costes de equipos se han incluido solos los referentes al material utilizado para realizar el proyecto. El servidor en el que se instalará el aplicativo ADGSO y las licencias de Oracle no se han incluido, dado que ambas las provee la Universidad.

Según las cifras establecidas en la hoja de cálculo del desglose presupuestario del proyecto, se puede determinar que el presupuesto total de este proyecto asciende a la cantidad de veinte mil cuatrocientos sesenta y dos (20.462 €) euros.

Leganés a 15 de Julio de 2011

El ingeniero proyectista

Fdo. Esther Herva Landeira

## 1.- Autor:

Esther Herva Landeira

## 2.- Departamento:

Departamento de Bases de Datos Avanzadas

## 3.- Descripción del Proyecto:

| | |
|---|---|
| - Titulo | **ADGSO- Aplicación con fines didácticos para la creación y carga de datos espaciales en Oracle** |
| - Duración (meses) | **5,5** |
| Tasa de costes Indirectos: | **20%** |

## 4.- Presupuesto total del Proyecto (valores en Euros):

Euros

## 5.- Desglose presupuestario (costes directos)

### PERSONAL

| Apellidos y nombre | N.I.F. (no rellenar - solo a titulo informativo) | Categoría | Dedicación (hombres mes) [a] | Coste hombre mes | Coste (Euro) | Firma de conformidad |
|---|---|---|---|---|---|---|
| | | Jefe de Proyecto | 0,2 | 5.600,00 | 1.120,00 | |
| | | Analista | 4,05 | 3.200,00 | 12.960,00 | |
| | | Diseñador | 0,25 | 2.600,00 | 650,00 | |
| | | Programador | 1 | 2.200,00 | 2.200,00 | |
| | | **Hombres mes** | **5,5** | **Total** | 16.930,00 | |

[a] 1 Hombre mes = 100 horas.

### EQUIPOS

| Descripción | Coste (Euro) | % Uso dedicado proyecto | Dedicación (meses) | Periodo de depreciación | Coste imputable [d] |
|---|---|---|---|---|---|
| Oracle Express Edition | 0,00 | 100 | 6 | 60 | 0,00 |
| Tomcat 6 | 0,00 | 100 | 6 | 60 | 0,00 |
| PC Intel® Core™ i3 550 4GB RAM 500GB | 499,00 | 100 | 6 | 60 | 49,90 |
| Portátil Intel® Core™ i3-350M 4GB RAM 640GB | 599,00 | 100 | 6 | 60 | 59,90 |
| Monitor LCD LED 2011x 50,8 cm (20'') | 119,00 | 100 | 6 | 60 | 11,90 |
| | | | | **Total** | 121,70 |

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

**A** = nº de meses desde la fecha de facturación en que el equipo es utilizado
**B** = periodo de depreciación (60 meses)
**C** = coste del equipo (sin IVA)
**D** = % del uso que se dedica al proyecto (habitualmente 100%)

### SUBCONTRATACIÓN DE TAREAS

| Descripción | Empresa | Coste imputable |
|---|---|---|
| | | |
| | | |
| | **Total** | 0,00 |

### OTROS COSTES DIRECTOS DEL PROYECTO [e]

| Descripción | Empresa | Costes imputable |
|---|---|---|
| | | |
| | | |
| | **Total** | 0,00 |

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo:

## 6.- Resumen de costes

| Presupuesto Costes Totales | **Presupuesto Costes Totales** |
|---|---|
| Personal | 16.930 |
| Amortización | 122 |
| Subcontratación de tareas | 0 |
| Costes de funcionamiento | 0 |
| Costes Indirectos | 3.410 |
| Total | 20.462 |

# Glosario

| | |
|---|---|
| API | *Application Programming Interface* |
| CAD | *Computer Aided design* |
| CAM | *Computer Aided Manufacturing* |
| CRM | *Customer Relationship Management* |
| CSS | *Cascading Style Sheets* |
| ERP | *Enterprise Resource Planning* |
| GML | *Geographic Markup Language* |
| HTTP | *HyperText Transfer Protocol* |
| IDE | *Integrated Development Environment* |
| JDBC | *Java Data Base Connectivity* |
| JDK | *Java Development Kit* |
| JSP | *JavaServer Pages* |
| JSF | *JavaServer Faces* |
| J2SE | *Java 2 Platform, Standard Edition* |
| LRS | *Linear Referencing System* |
| MVC | *Modelo Vista Controlador* |
| OCI | *Oracle Call Interface* |
| OGC | *Open Geospatial Consortium* |
| OTN | *Oracle Technology Network* |
| SIG | *Sitema de Información Geográfica* |
| SGBD | *Sistema Gestor de Bases de Datos* |
| SGBDR | *Sistema Gestor de Bases de Datos Relacionales* |
| SDK | *Software DevelopmentKit* |
| SOA | *Service Oriented Architecture* |
| SQL | *Structured Query Language* |
| URL | *Uniform Resource Locator* |
| WAR | *Web Archives Repository* |
| WKB | *Well Known Binary* |
| WKT | *Well Known Text* |
| W3C | *World Wide Web Consortium* |
| XML | *Extensible Markup Language* |

# Referencias


[AENOR, 2004] AENOR. '*Asociación Española de normalización y certificación, Norma UNE 139803: Requisitos de accesibilidad para contenidos en la Web www.aenor.es, 2004*'.
Disponible [Internet]:
http://www.inteco.es/Accesibilidad/Normativa_1/Descarga/DescargaUNE_139803
, Accedido en 2011.

[ARCGIS] http://help.arcgis.com, Accedido en 2011.

[AUTODESK] http://www.autodesk.es, Accedido en 2011.

[BENTLEY] http://www.bentley.com, Accedido en 2011.

[ESRI] http://www.esri.es/es/, Accedido en 2011.

[GEOMEDIA] http://www.ambaoyasoc.com.ar/prod06.php, Accedido en 2011.

[IEEE, 1987] IEEE Computer Society. '*IEEE Standard for Software Project Management Plans (IEEE 1058.1-1987)*'.

[JSF] http://javaserverfaces.java.net, Accedido en 2011.

[MAB+, 2009] Chuck Murray,Dan Abugov, Nicole Alexander, Bruce Blackwell, Raja Chatterjee, Dan Geringer, Mike Horhammer, Ying Hu, Baris Kazar, Ravi Kothuri, Siva Ravada, Jack Wang, Ji Yang*: 'Oracle Spatial Developer's Guide11g Release 1 (11.1)'* (Oracle, 2009).

Disponible [Internet]:
http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28400.pdf.

[MPY, 2010] Chuck Murray, Joao Paiva, L.J. Qian, Ji Yang: '*User's Guide for Oracle MapViewer 11g*'(Oracle, 2010).
Disponible [Internet]:
http://download.oracle.com/docs/cd/E17904_01/web.1111/e10145.pdf

[KGE, 2007] Ravi Kothuri, Albert Godfrind, Euro Beinat: 'Pro Oracle Spatial for Oracle Database 11g'(Apress, 2007, 1st edn.)

[Peñ, 2088] Juan Peña Llopis: 'Sistemas de Información Geográfica aplicados a la gestión del territorio' (Editorial Club Universitario, 2008, 3ª Edición)

[RG, 2006] Andrea Rodríguez, Francisco Godoy. '*Bases de Datos Espaciales*'. Departamento de Ingeniería Informática y Ciéncias de la Computación, Univerdidad Concepcion.
Disponible [Internet]: http://www.inf.udec.cl/~andrea/cursos.htmlMicrosoft.

[SIG]
http://es.wikipedia.org/wiki/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica, Accedido en 2011.

[TOMCAT] http://wiki.apache.org/tomcat/FrontPage, Accedido en 2011.

[VB, 2008] José Luis Vicente González, Virginia Behm Chang. Consulta, Edición y Análisis Espacial con ArcGIS 9.2. Consejería de Medio Ambiente, Junta de Castilla y León (2008)